COMPUTER SIMULATED VISUAL AND TACTILE FEEDBACK AS AN AID TO MANIPULATOR AND VEHICLE CONTROL

by
Calvin McCoy Winey III
S.B. MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE IN
MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1981

Signature of Author_____
Department of Mechanical Engineering
May 8,1981

Certified by_____
Thomas B. Sheridan
Thesis Supervisor

Accepted by_____
Warren M. Rohsenow
Chairman, Department Committee

COMPUTER SIMULATED VISUAL AND TACTILE FEEDBACK AS AN AID TO MANIPULATOR AND VEHICLE CONTROL

by
CALVIN McCOY WINEY III

Submitted to the Department of Mechanical Engineering on May 8, 1981 in partial fulfilment of the requirements for the Degree of Master of Science in Mechanical Engineering

ABSTRACT

A computer graphic simulation of a seven degree-of-freedom slave manipulator controlled by an actual master was developed. An electronically coupled E-2 manipulator had previously been interfaced to a PDP-11/34 by K. Tani, allowing the computer to sense and control each degree of freedom independently. The simulated manipulator was capable of moving an arbitrarly shaped object and sensing a force in an arbitrary direction with no actual object or force existing. The simulated manipulator could also be attached to a simulated vehicle capable of motion with six degrees-of-freedom. The vehicle simulation is currently being used in conjunction with dynamic simulations developed by H. Kazerooni to test different types of dynamic controllers for submarines. Shadows, multiple views and proximity indicators were evaluated to determine their effectiveness in giving depth information. The results indicated that these aids are useful. Subjects felt that shadows gave the best perception of the environment, but found isometric views easiest to use on the tasks performed. This type of simulation appears to be realistic and adaptable to a multitude of applications.

Thesis Supervisor: Thomas Sheridan
Title: Professor of Engineering and Applied Psychology

ACKNOWLEDGEMENTS

I wish to thank Professor Thomas Sheridan for his many words of advice and encouragement. I appreciate the opportunity to work in the Man-Machine Systems Laboratory, it has been a most interesting and educational experience.

I would like to thank my many friends at the Man-Machine Systems Lab for making it an enjoyable place to work. Special thanks must go to my friend and co-reseacher, Homayoon Kazerooni, who developed the vehicle dynamics simulation and helped me interface it with the graphic simulation. I would also like to thank Don Fyler for his suggestions on improving the simulation, Dana Yoeger for helping me to understand the computer system and Ahmet Buharali for helping to maintain the computer. A word of thanks must go to all the people who put time into performing my experiments at a very busy time in the semester.

Finally, I wish to thank my fiancee, Deborah Darago, for her help on the statistical analysis of the experments, for proofreading this document and for her patience and understanding during the many hours spent working on this project.

This work was supported by the Office of Naval Research, Contract N00014-17-C-0256, monitored by Mr. G Malecki, Engineering Psychology Programs.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

INTRODUCTION

In the past several years, develpoments in the electronics industry have made mini-computers extremely small, powerful and inexpensive. Microprocessors are now being incorporated in machinery ranging from large scale production equipment to household dishwashers. As this trend continues, more effort will be put into the use of the computer to aid the human operator. Automobiles are already on the market which use microprocessors to control automobile function and relay failure information to the driver. As computers become more common, the question will arise as to how they can best serve the operator.

Artificial Intelligence Versus Supervisory Control

The use of computers to aid human operators can be divided into two catagories: artificial intelligence and supervisory control. The major difference between the two approaches is in the manner in which the computer interacts with the human operator. Artificial intelligence (A. I.) attempts to give the computer maximum intelligence and to replace all operator functions by the computer. Supervisory control acknowledges that the operator has certain abilities. It attempts to use those talents and supplement those which are lacking.

The task of removing a bolt from an undersea structure emphasizes the differences between these two approaches.

The A. I. approach might require the computer to differentiate between the pilings and the surroundings, to be able tell a bolt from a barnacle and be able to select the proper bolt. It must also be able to determine the proper angle at which to turn the nut, be able to select the proper tool to fit that nut and cope with the possability of the nut being damaged. The supervisory control approach would rely on the operator to find the bolt, while possibly aiding with image enhancement. The operator would determine the proper tools and condition of the bolt. If the vehicle were moving, the computer might aid the operator by compensating for that motion. The computer would then remove the bolt after the operator showed it the proper orientation and described the desired motion.

The A. I. approach becomes necessary when circumstances make operator interaction impossable. Its major drawback is that it requires extensive programming in order to cope with all possible contingencies. The more variable the environment; the more complex the required programming becomes. The supervisory approach achieves economy by taking advantage of the operator's abilities (experience and intuitive skills) while the computer supplies memory, accurate position control, speed and repeatability.

Computer Generation of Operator Feedback

Since supervisory control relys on operator inter-

action, one important way in which a computer can aid an operator is to improve the operator's knowledge of his environment. Feedback can take many forms, though humans rely most heavily on their tactile, visual and auditory senses. Improved feedback is important for several reasons. Supplying the operator with more processed information leaves the operator more time to dedicate to the task. Any information can be related to the operator by a set of numbers, however the some types of feedback are more easily assimulated than others. The orientation of a vehicle can be completely described by a set of angles, however a display of the vehicle is more informative even though it is less accurate. Feedback may exist, but it may be of poor quality. In this case, the computer could be used to improve rather than create feedback.

Several forms of feedback can be used simultaniously to reinforce the operator's perception of the environment. For example, combining tactile with visual feedback may be a better aid than either tactile or visual feedback alone. A single type of feedback may not be best suited for all tasks. The use of several different types of feedback could allow the operator to select the type of feedback he preferred for each type of task.

Simulation of Master-Slave Manipulators

The basic part of this research was the development

of a computer graphic simulation of a master-slave manipulator. The simulation was controlled by an E-2 master manipulator which had previously been interfaced to a PDP11/34 computer by K. Tani (7). This interface allowed the computer to sense and control each of the seven degrees of freedom of the manipulator independently. Part of the simulation included the development of an environment within the computer which the simulated manipulator could interact with. The simulated manipulator was capable of moving an arbitrarly shaped object about in three-dimensional space and simulating force-feedback in an arbitrary direction. Force was felt when the manipulator grasped an object or touched a predefined surface. The simulated manipulator could also be attached to a vehicle capable of motion with six degrees-of-freedom. The vehicle simulation is currently being used by H. Kazerooni to test various types of dynamic controllers for small underwater vehicles.

Depth Perception From Two-Dimensional Images

One of the difficulties common to both generating graphics of a three-dimensional system and to performing manipulation via closed-circuit television is the lack of ability to perceive three dimensions. The method of displaying depth would appear to be particularly critical in applications such as manipulation which require physical motion to be coordinated to visual input. Tasks such as grasping an

object from a moving vehicle require the ability to quickly integrate the two-dimensional image plus the depth cues into three-dimensional motion. If the information cannot be assimilate quickly, the object will have moved relative to the operator by the time its position has been determined. A close spatial relationship between the display and the real world would appear to make interpreting the display easier.

Shadows, multiple views and proximity indicators were tested to determine their effectiveness as depth cues. Each depth indicator was chosen for exemplifying a particular attribute. The shadow was chosen because it is the most familiar depth cue and has a strong spatial relationship. Its drawbacks are that it compresses picture information and there is the possablity that several shadows near one another may make interpretation difficult. The front and side isometric projections normally used in mechanical drawings give depth information more clearly than the shadow, but determination of depth from the side view may cause some coordination problems. Both of these depth indicators require extensive prior knowledge of the environment. The proximity indicator which measures the distance between the manipulator tongs and the desired destination could be displayed with little prior knowledge of the environment. In practice, proximity could be determined by a sonar mounted on the tongs. The proximity indicator was displayed along

with a front view of the manipulator. The proximity indicator, with no other display, was used as a control.

APPLICATIONS OF COMPUTER GRAPHIC MANIPULATOR SIMULATIONS

Simulation of Various Operating Conditions

A realistic simulation can be of value both for expermentation and for operator training by simulating environments which can not be easily created in a laboratory. The viscosity and the change in the relationship between weight and mass associated with underwater work can be simulated without requiring tanks of water or undersea manipulators. The zero-gravity conditions associated work in space such as the space shuttle can be easily simulated by a computer, but would be very difficult to obtain in a laboratory by any other means.

By using a computer to control a manipulator, it is possable to vary the properties of that manipulator so that it can be used to simulate many types of manipulators. Moments of inertia can be changed to make a light hot-room manipulator behave like a massive industrial manipulator. Several degrees-of-freedom of a seven degree-of-freedom manipulator can be locked to simulate a less flexible manipulator.

Testing of Control Systems

Building a control system for a vehicle can be an expensive and time consuming undertaking. Propulsion units

need to be modified and sensors need to be installed. With a new controllor there is always the risk of instablity and failure which could result in damaged hardware. When creating a control system for a one-of-kind vehicle, the money and time lost in a failure could make improvement or addition of a control system prohibitive. Using a computer simulation would allow the prototype controllor to be changed quickly and easily. Propulsion and sensor configurations could be tested without the need for expensive hardware. Failure of a computer simulation generally involves essentially no risk. Therefore, a simulation can be run during an instability to collect additional data on the failure with out the risk of hardware damage.

Supplemention of Operator Feedback

Due to the high cost of using humans directly, unmanned submersibles are used to inspect and repair offshore oil rigs in the North Sea. To avoid tethering problems, communicatons to the operator on the surface can be via accoustic link. One difficulty with an accoustic link is that it is only capable of low bit rate transmission. The bit rate is the product of the frame rate, number of bits of gray scale and the number of pixels (resolution). A 200 by 200 pixel picture with 5 bits of grey and a frame rate at the flicker limit of 15 frames per second requires transmitting 3 million bits per second. To the operator who is watching

the work via a television picture sent through an accoustic link, this means he receives a very degraded picture (3). Since the manipulator position can be completely defined by knowing each angle of the seven degrees-of-freedom to 16 bits, these 118 bits can be transmitted freqently allowing the simulation to be updated frequently. By superimposing a rapidly updated simulation of the manipulator on a slowly updated but high resolution television picture, data transmission can be optimized such that the moving portion of the display is refreshed frequently while the static visual background is of good quality.

In the case of poor visability, a simulation could be used to generate or enhance the view of the surroundings. If part of the environment was known in advance and stored in the computer, it could be displayed on the simulation as soon as the operator established enough reference points to locate and orient the environment relative to the operator. Objects could also be inputted into the display by feeling about and recording points of contact. The points of contact could then be analyzed to determine the location of surfaces and edges.

Rehearsal

When an operator is required perform a dangerous or delicate task where a mistake might harm the operator, the equipment or the task, it might be desirable for the oper-

ator to be able to practice the task first before actually performing it. If a realistic simulation was available, the operator could practice the task on the simulation until he felt confident to actually execute the task. If the manipulator were computer controlled, the operator could perform the task on the simulation until he performed the task perfectly. The computer could moniter each practice run. When the operator was satisfied with a run, he could tell the computer execute that run. The computer would then duplicate the previous motion (6).

Humanizing Man-Computer Interaction

It has been suggested that a standard master manipulator or a smaller table-top version be used as a means of communicating with a computer. The manipulator used in these experiments had seven degrees-of-freedom which allowed the tongs to be moved to any position and orientation within range. It was also capable of force-feedback which allowed it to communicate contact with an object to the operator. One of the difficulties with CAD systems is developing the ability to input three-dimensional data. A manipulator could be used as a three dimensional digitizer. The spatial quality of the manipulator might help in inputting points which were three-dimensional in nature. The manipulator could send force-feedback when the operator touched a point or line in order to aid the operator in locating references. If the

desired points were all on a given plane, computer control could be used to restrict movement to that plane. When the desired object had been inputted, it could be easily examined by grasping it and rotating it as if it were in one's hand.

The manipulator could also be used in a more abstract way. Data could be encoded with spatial, tactile or physical properties. A particular point might be hard or soft, heavy or light, or sticky or slippery. This might be helpful in aiding the operator to select a particular type of data while searching through a data space. It might also help him to notice small differences between attributes. If he were looking for a particular type of data, the computer could use force-feedback to "push" him in the right direction (6).

DISPLAY THEORY

Data Storage

A schematic of the manipulator and vehicle with the definitions of the degrees-of-freedom is shown in figure 1. The arm, submarine and object were stored in standard point-connectivity data form. The arm was broken into three distinct portions, the shoulder, forearm and tongs. Each section of display was treated as a separate entity. The data base for each section was stored in an unrotated reference frame with the center of rotation located at the

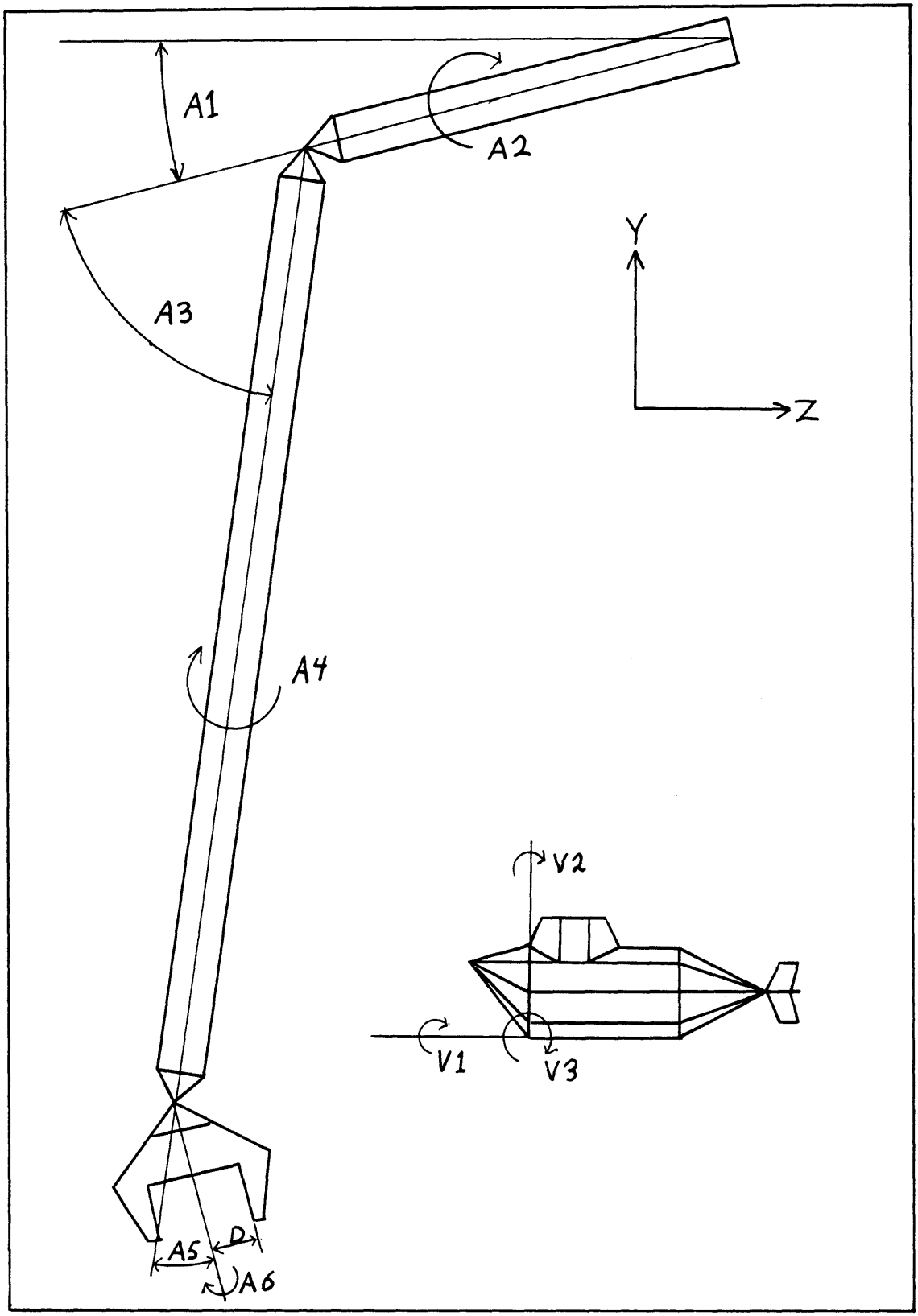

Figure 1 Manipulator and Vehicle Angles

origin. The vehicle was stored so that forward motion of the unrotated vehicle was in the negative z direction. Each section of the arm extended from the origin along the negative z-axis. This reference frame was the same as that of the display terminal. In this right-handed coordinate system, the x-axis was to the right, the y-axis was upwards and the z-axis was out of the screen. The origin was at the center of the screen. Each element had a corresponding rotation matrix containing the transformations required to move the element from the reference frame to its desired location. Objects which could be manipulated also had a set of touching conditions which were defined in the reference frame.

Translations and Rotations

The multiple rotations required to display the arm were most easily calculated in matrix form. The matrix elements were fed directly into the display processor's hardware matrix multiplier. All rotations were based on global coordinates. A translation Tx in the X direction is given by:

$$X' = X + Tx$$

In three dimensions this can be expressed in matrix form as:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where the ones in the coordinate matrices satisfy the matrix algebra needed to add the constant Tn. This matrix form can be abbreviated as:

$$\overline{X}' = \overline{T}\ \overline{X}$$

$\overline{T}$ is called the translation matrix. Transformation matrices can also be formed for rotations in the same manner. The rotation matrix $\overline{Rx}$ for a rotation of an angle A about the x-axis is:

$$\overline{Rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos A & -\sin A & 0 \\ 0 & \sin A & \cos A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation matrices about the y and z-axis are given by:

$$\overline{Ry} = \begin{bmatrix} \cos A & 0 & \sin A & 0 \\ 0 & 1 & 0 & 0 \\ -\sin A & 0 & \cos A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \overline{Rz} = \begin{bmatrix} \cos A & -\sin A & 0 & 0 \\ \sin A & \cos A & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These rotation matrixes are for rotation about the origin. To rotate about an arbitrary point, one must translate the desired center of rotation to the origin, perform the rotation, then translate the center of rotation back to its original position (4,5).

A series of transformations can be reduced to a single transformation matrix by matrix multiplication. Since matrix multiplication is associative but is not commuta-

tive, the order in which transformatioms are performed is important. Rotation about the x-axis and then about the z-axis is not the same as rotation about the z-axis and then about the x-axis. This is most easily seen by the example illustrated in figure 2. If one holds one's right arm out to the side with the palm facing downward and a coordinate system is established so that the y-axis is upward and the x-axis extends to the right parallel to ones arm, then the z-axis extends out behind. If the forearm is rotated 90 degrees about the x-axis, then 90 degrees about the y-axis, the forearm is pointed forward and the palm is facing to the left. If the forearm is first rotated 90 degrees about the y-axis, then 90 degrees about the x-axis, the forearm is facing upward and the palm is facing foreward. Transformation matrices occur before the coordinate matrices and are sequenced in the order in which they occur from right to left as follows:

$$\overline{X}' = \overline{A}'\overline{X} = \overline{A}n \ \ldots \ \overline{A}3 \ \overline{A}2 \ \overline{A}1 \ \overline{X}$$

In this simulation it was necessary to be able to perform the reverse coordinate transformation. This requires taking the inverse of the transformation matrix. The inverse of $\overline{A}$ is found by:

$$\overline{A}\text{-}1 = \mathrm{adj}(\mathrm{transpose}(\overline{A}))/\det(\overline{A})$$

For a 4x4 matrix, computing the adjoint requires finding

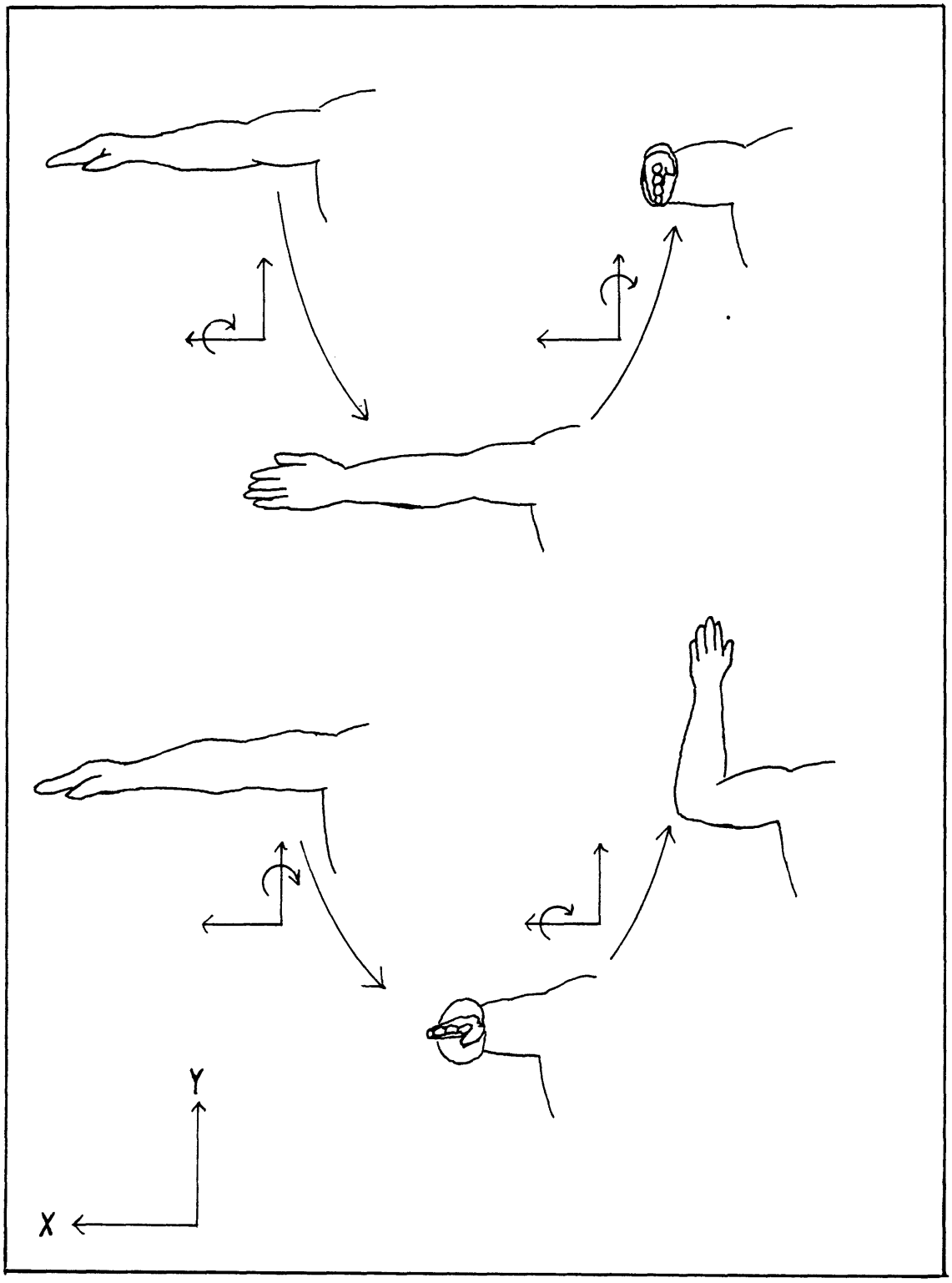

Figure 2 Effects of Order of Rotation

determinants of 16 3x3 matrices, each of which requires 5 additions and 6 multiplications, for a total of 176 operations. Since the bottom row of the transformation matrix is only used to satisfy the matrix algebra and contains no information, the transformation matrix can be partitioned into a 3x3 rotation matrix and a 1x3 translation matrix:

$$\left[\begin{array}{c|c} R & T \\ \hline 0 & 1 \end{array}\right]$$

The coordinate transformation is then given by:

$$\overline{X}' = \overline{R}\overline{X} + \overline{T}$$

This can be solved for X to give:

$$\overline{X} = \overline{R}^{-1}\,\overline{X}' - \overline{R}^{-1}\,\overline{T}$$

Computing the adjoint of a 3x3 matrix requires finding the determinants of 9 2x2 matrices, each of which require 1 addition and 2 multiplications for a total of only 27 operations. The new transformation matrix can be reconstructed by rejoining the new rotation and translation matrices as:

$$\left[\begin{array}{c|c} R^{-1} & R^{-1}\,T \\ \hline 0 & 1 \end{array}\right]$$

For the general rotation matrix,

$$\begin{bmatrix} XX & XY & XZ & XT \\ YX & YY & YZ & YT \\ ZX & ZY & ZZ & ZT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the components of the inverse matrix,

$$\begin{bmatrix} RXX & RXY & RXZ & RXT \\ RYX & RYY & RYZ & RYT \\ RZX & RZY & RZZ & RZT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

are fould to be:

RXX=(YYZZ-ZYYZ)/Det RXY=(ZYXZ-XYZZ)/Det
RXZ=(XYYZ-YYXZ)/Det RXT=-RXXXT-RXYYT-RXZZT
RYX=(YZZX-YXZZ)/Det RYY=(XXZZ-XZZX)/Det
RYZ=(YXXZ-YZXX)/Det RYT=-RYXXT-RYYYT-RYZZT
RZX=(YXZY-YYZX)/Det RZY=(XYZX-ZYXX)/Det
RZZ=(XXYY-XYYX)/Det RZT=-RZXXT-RZYYT-RZZZT

where det is the determinant of the rotation matrix and is given by:

Det=XXYYZZ+XYYZZX+XZYXZY-ZXYYXZ-ZYYZXX-ZZYXXY

The determinate of a rotation matrix is formally one, however it was calculated to compensate for roundoff error by the computer.

Shadow Generation

If the source of illumination is far enough from an object that rays passing through different points can be considered parallel, then its shadow can be represented by matrix notation. If the source of illumination is directly overhead, then for a shadow cast on a horizantal surface

the y-coordinate of every point in the shadow is at the height of the surface Ys and x and z values remain unchanged. The transformation matrix for this shadow is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & Ys \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If the surface can be expressed as a linear function of x and z as given by:

$$y=Ax+Bz+C$$

then the transformation matrix for a shadow cast on this surface is given by:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ A & 0 & B & C \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is also possable to accomodate light coming in from an arbitrary angle. Suppose the light is from an angle (a) from the y-axis, as measured about the z-axis and the surface is horizantal at Ys. The x component of the shadow remains the same as that of the object, the y component is at the surface value Ys. The z value is determined from the distance between the surface and the object and the angle of illumination.

$$X'=X$$
$$Y'=Ys$$
$$Z'=Z-(Y-Ys)Tan(a)$$

The transformation matrix is then:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & Ys \\ 0 & -Tan(a) & 1 & YsTan(a) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shadows on different surfaces and illumination from different directions can be handled in a similar manner.

Inertia

It is hoped future vehicle simulations will include manipulator-vehicle interactions such as those which occur when manipulating a massive object from a small, light vehicle. This will require determining the moment of inertia of the manipulator for any position. If the centroidal moment of inertia $\overline{I}c$ is known for each unrotated section of the manipulator, then the moment of inertia for any manipulator configuration can be determined (2). First the inertance must be calculated for the proper orientation of each segment of the manipulator. If a segment is described by a rotation matrix $\overline{R}$, were $\overline{R}$ is the rotation portion of the transformation matrix, then the inertance $\overline{I}r$ for the rotated segment is given by:

$$\overline{I}r=\overline{R}\overline{I}c\overline{R}^{T}$$

The final moment of inertia $\overline{I}'$ about the shoulder can the be found by applying the parallel axis theorem. If Xt, Yt and Zt are the distances from the shoulder to the centroid

of the segment in question and M is the mass of the section, then the inertance is:

$$\overline{I}' = \overline{R}\overline{I}_c\overline{R}^T + M \begin{bmatrix} Yt2+Zt2 & -XtYt & -XtZt \\ -XtYt & Xt2+Zt2 & -YtZt \\ -XtZt & -YtZt & Xt2+Yt2 \end{bmatrix}$$

If $\overline{X}c$ is the vector describing the location of the centroid in the unrotated reference frame, then the vector $\overline{X}t$ is found to be:

$$\overline{X}t = \overline{R}\overline{X}c$$

Since the manipulator is oriented along the z-axis, if the manipulator is relatively symmetric about the z-axis, Xc and Yc are zero, in which case the elements of Xt are simply:

$$Xt = XZXc + XT$$
$$Yt = YZYc + YT$$
$$Zt = ZZZc + ZT$$

Global and Body Coordinates

The orientation of a vehicle is normally expressed in the body-referenced Euler angles yaw, pitch and roll. The order of rotation is yaw, followed by pitch and finally roll. The reason for this convention is that forces on a body are generally invarient with respect to changes in orientation when they are described by body coordinates. The dynamic programs developed by H. Kazerooni are based on body centered coordinates. Since the display processor is based on global coordinates, a method of transforming from body centered to global coordinates was needed.

Examination of motions of the vehicle reveals that the rotations yaw, pitch and roll in body coordinates are the same as roll, followed by pitch and then yaw in global coordinates. Before any rotation, global and body coordinates are equivalant (Figure 3). Roll about global and body axis is therefore the same. After this rotation, pitch about global coordinates performs the same function as it did in body coordinates. Final yaw performed in global coordinates after these transformations is still the the same as yaw performed first in body coordinates.

Generation of Force-Feedback

In normal manipulator operation, the only information transfer between our master and slave manipulator is positional information. The control interface allows positional information to be transferred between computer and manipulator. Since there is no means of directly sending force information to the manipulator, generation of force-feedback involves encoding force information into a positional signal.

To understand computer generated force-feedback, one should first understand the operation of the manipulator in master-slave mode. Figure 4 shows the configuration of the manipulator in this mode. Each manipulator is made up of a set of servomotors, each of which is directly coupled to a potentiometer. Each servomotor is driven by an amplifier

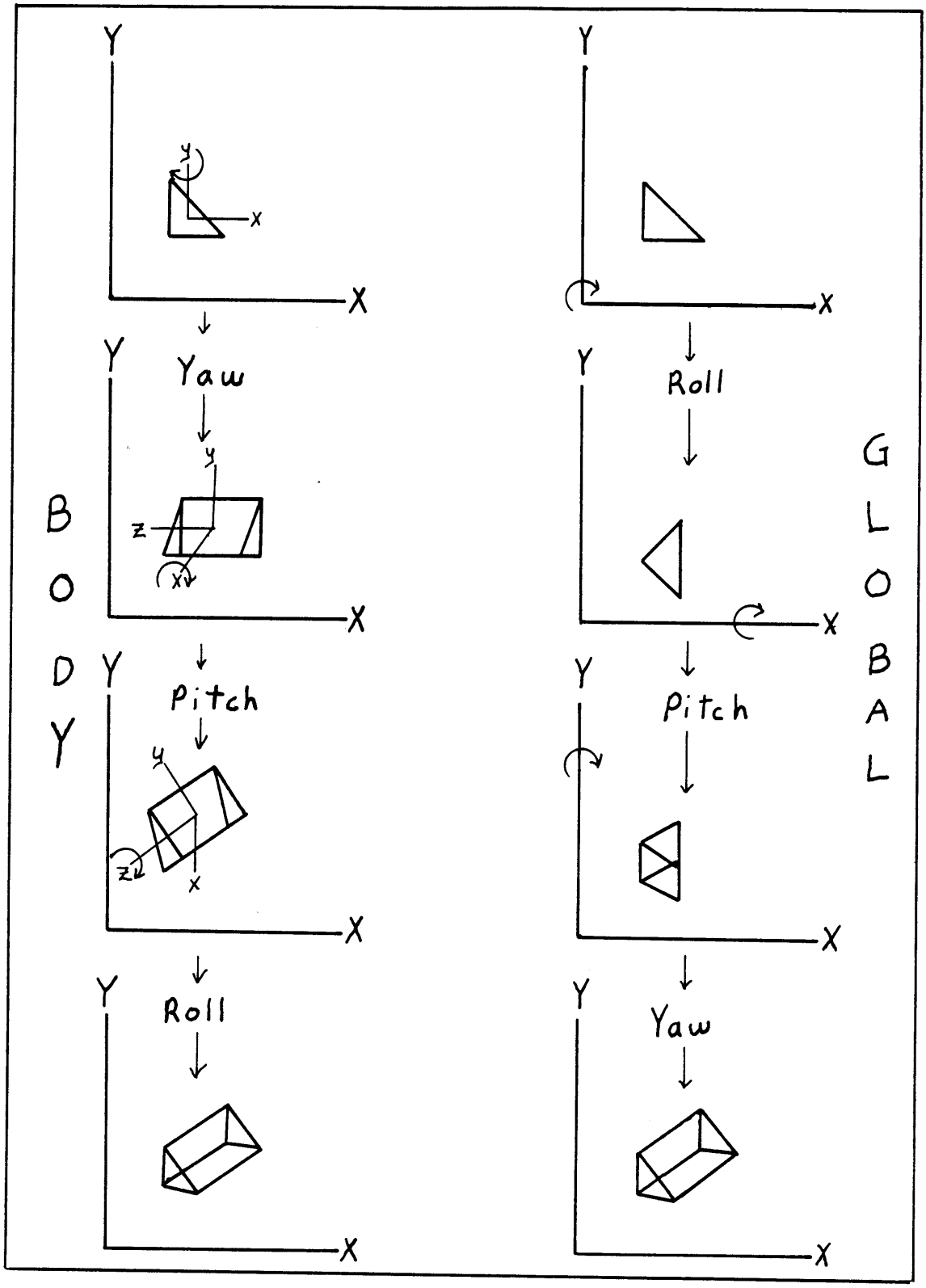

Figure 3 Equivalent Body and Global Rotations

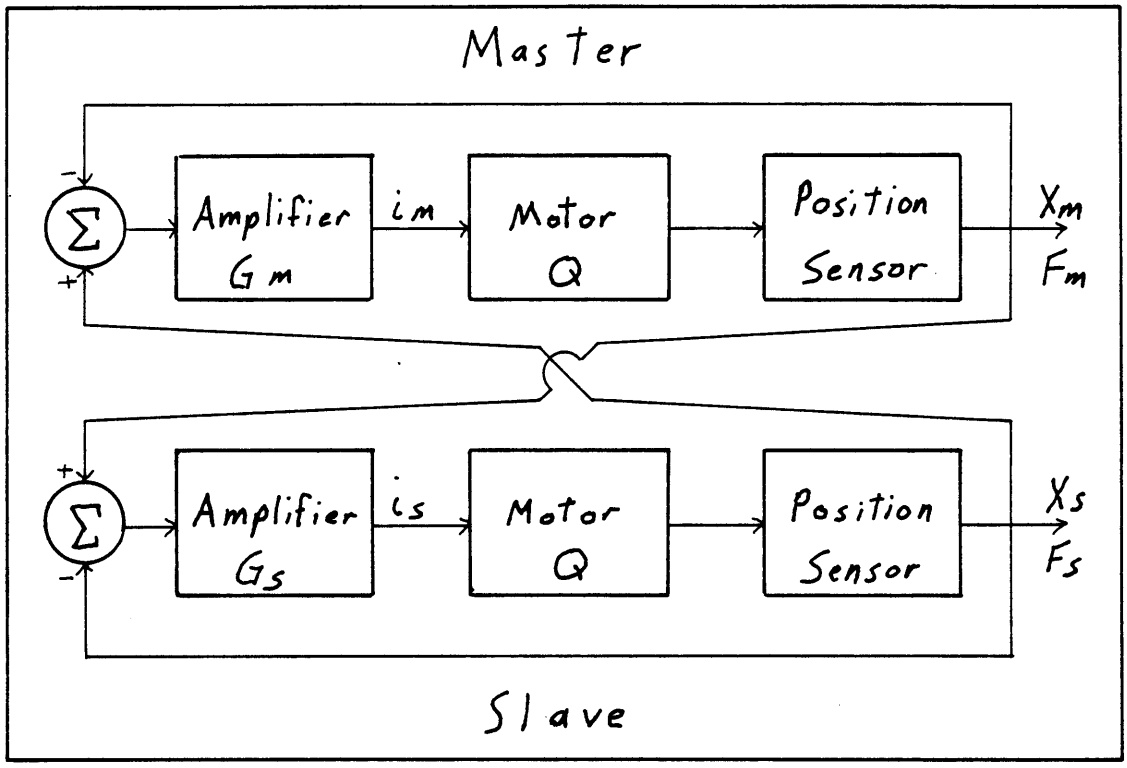

Figure 4 Master-Slave Manipulator Block Diagram

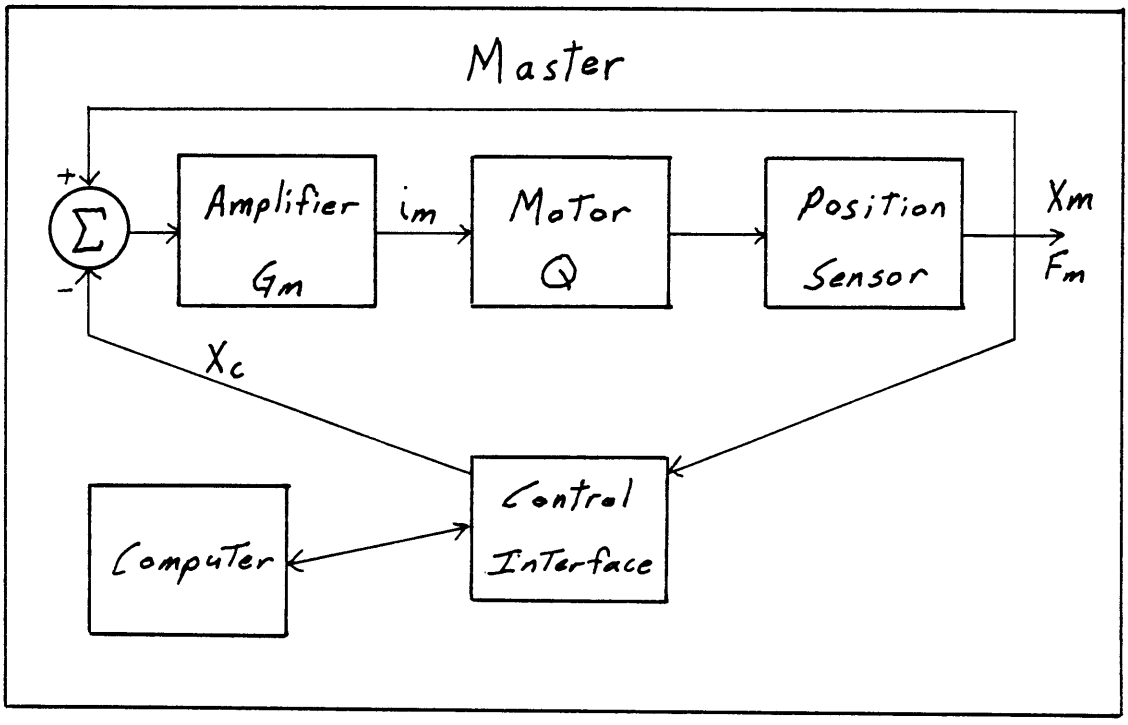

Figure 5 Computer Controlled Master Block Diagram

that outputs a current which is proportional to the difference between the master and slave positions. If Xm and Xs are the master and slave positions respectively, and Gm and Gs are the gains of the amplifiers associated with the master and slave, then the currents input to the servomotors are given by:

$$is=Gs(Xm-Xs)$$
$$im=Gm(Xs-Xm)$$

If Q represents the gain of the servomotors in converting current into force, then:

$$Fs=QGs(Xm-Xs) \quad (1)$$
$$Fm=QGm(Xs-Xm) \quad (2)$$

Combining equations 1 and 2 yeilds:

$$Fs=-FmGs/Gm \quad (3)$$

or force on the master is directly proportional to the force on the slave.

Since only positional information is transferred between the master and slave, how that information relates to forces on master and slave needs to be considered. If the slave is pressing on a linear elastic surface, the force on the slave is given by:

$$Fs=K\Delta X \quad (4)$$

which is the conventional spring law where ΔX is the displacement of the surface and K is the spring constant. Combining equations 3 and 4 gives the resulting force felt through the master:

$$Fm=-K\Delta XGm/Gs$$

Combining equations 1 and 4 and solving for (Xs-Xm) yields the offset between master and slave position dX for a given displacement of the surface ΔX

$$dX=(Xs-Xm)=K\Delta X/(QGs) \qquad (5)$$

Using the above information, the input to the master manipulator required for computer generated force feedback can be determined.

Figure 5 shows the configuration of the master manipulator under computer control. It is the same as the master-slave configuration except that the slave position is replaced by a computer input. If the slave position Xs is replaced by computer input Xc, equation 5 becomes:

$$dX=(Xc-Xm)=K\Delta X/(QGs) \qquad (6)$$

In a computer simulation, the penetration into the surface is given by the difference between a reference point at the surface and the position of the master (eq. 7)

$$X=Xref-Xm \qquad (7)$$

Equation 6 can be rewritten using equation 7 as:

$$Xc=Xm+K(Xref-Xm)/(QGs) \qquad (8)$$

This equation gives the computer input Xc required to generate force feedback proportional to the stiffness of the surface and the penetration into the surface. If the surface is infinitely deformable (K=0), then Xc=Xm and the computer allows the manipulator to move freely. If the surface is com-

pletely rigid (K/(QGs=1), then Xc=Xref. Hence the computer will not allow the manipulator to penetrate the reference plane.

Since simulating a force requires directing the manipulator to move to a position other than its current position, the need arises to calculate the manipulator angles corresponding to that position. Our trial of the concept used a linear interpolation method. When the manipulator first touched a surface, its angular position was saved as a reference. As long as the manipulator remained within the surface, the desired manipulator position was calculated to be a weighted mean of the current position and the reference position. This approach proved the concept but had two problems. First, since the program cycled in discrete time steps, it was possable for the manipulator to have penetrated well into the surface before the computer realized that penetration had occurred. The result was a reference position which is within the surface rather than on the surface. When the manipulator was withdrawn, the surface seemed tacky because the computer attempted to pull the the manipulator back to the reference rather than releasing the manipulator. The second problem concerned the direction of the force-feedback. The force was generated was along a vector defined by the reference position and current manipulator position, instead of normal to the surface. The

result was similar to that of a rubberband being attached between the manipulator and the reference position.

Resolved Motion Rate Control

Both these problems were overcome by developing a transformation to go from cartesian coordinates to the manipulator's multiple angle coordinate system. There are several problems associated with directly solving the transformation equations in term of manipulator angles. First, there are three transformation equations and six unknown angles. A value for three angles must be assumed to define a unique solution for the remaining three angles. Careful inspection of the function of the six angles (figure 1) reveals that, due to the difference in lever arm associated with rotation of the tongs about the shoulder and rotation about the wrist, angles A1, A2 and A3 control hand position and have only a small effect on hand orientation. A4, A5, and A6 control hand orientation and have little effect on hand position. Since differences between manipulator position and computer input position are small, A4, A5 and A6 can be assumed to be constant at their current values while changes in A1, A2 and A3 are used to position the manipulator.

The second problem is in solving the equations describing manipulator position for input angles, since arcsine and arccosine are not monotonic functions. Although an exact solution is possable, the simplest method is Resolved Motion

Rate Control (RMRC) proposed by D. Whitney (8), and successfully implimented in our manipulator by K. Tani (7). If

$$X=f(A)$$

where X is the position vector of the tip of the manipulator tongs and A is a vector made up of manipulator joint angles, then the differential of X is:

$$dX=JdA$$

where J(0) is the Jacobian of X given by:

$$\begin{bmatrix} dX/dA1 & dX/dA2 & dX/dA3 \\ dY/dA1 & dY/dA2 & dY/dA3 \\ dZ/dA1 & dZ/dA2 & dZ/dA3 \end{bmatrix}$$

Therefore, for incremental change in angle ΔA, the incremental change in position ΔX is given by:

$$\Delta X=J\Delta A \qquad (9)$$

For a given angular position, equation 9 can be solved for the incremental change in angles associated with a change of position:

$$\Delta A=J^{-1}\Delta X \qquad (10)$$

The new angular position A' of the manipulator can be determined by adding the incremental change in angle (equation 10) to the current manipulator position:

$$A'=A+J^{-1}\Delta X$$

Control of the Manipulator

To avoid striking objects with the slave while watching the display monitor, the slave arm was held in a fixed

position when the manipulator was used under control. The computer needed to allow only the master manipulator to move freely, or generate a force on the master while holding the slave fixed. Under computer control, the positions of the master and slave manipulator are set by the control interface. Deviations from these positions require applying appropriate forces to the manipulators. For the master manipulator to move freely under computer control, the computer must sense the current manipulator position, then feed that position to the control interface. Since the control program runs with a discrete cycle time, there is a period when the manipulator must be deflected from the position specified by the control interface. During this period, a force must be applied to the manipulator to maintain the deflection while waiting for the computer to realize the change in position and update the control interface. This results in the manipulator being quite stiff. Simulating the tachometer-forward loop used in master-slave operation helps alleviate the problem. If the current manipulator position is given by Xm and the last position is given by Xl, then the velocity of the manipulator is proportional to the difference between these two positions:

$$V \sim Xm - Xl$$

In simulating tach-forward loop signal, a position correction proportional to the velocity of the manipulator is added

to the new manipulator position. If the degree of tach forward is F, then the signal sent to the control interface Xc is:

$$Xc=Xm+F(Xm-X1)$$

A tach forward coefficient of .6-.7 worked well. A larger value generally lead to instability. K. Tani (7) found the inclusion of an acceleration term slightly beneficial, however it seemed to be of little value in this application.

A short cycle time was important for smooth operation. 20 to 30 milliseconds worked quite well and cycle times of up to 60 milliseconds were tolerable. Constant cycle time was also important. With these considerations in mind, the program which controlled the manipulator was separated from the program which performed the graphics and simulation. The program was slaved to a clock for even cycle time.

Object Motion

The simplest form of object motion is a change of position with no rotation. This can only be used with objects which are incapable of rotation such as sliding switches, or with objects which are symmetric about every axis such as spheres. The latter assertion holds since a sphere's profile does not vary with orientation and hence it may be displayed knowing only positional information. To move such an object requires only that the center of the

object translates along with the tongs.

Motion of more complex objects requires both positional and rotational information. Since the transformation matrix for the tongs is already known, the most efficient means of moving an object would be to use the same transformation matrix. This requires that the data base for the tongs and the object be in the same reference frame. When the tongs grip the object, the object is moved to the same reference frame as the tongs by applying the inverse of the tong transformation. The tongs and object can then be treated as a single entity. They can then be moved with the tong transformation until such time as tongs release the object.

Touching Conditions

Determining whether an object has been grasped can be broken down into two problems. First, the object must be between the tongs. Secondly the tongs must be closed enough to hold the object. This second condition requires that the tongs are closed to a size which is smaller than the width of the object measured in the direction of the normal to the jaws of the tongs.

The simplest method of determining whether an object has been gripped is to use spherical touching conditions. This requires that a spherical region of sensitivity (contact point) be defined corresponding to the size and posi-

tion of the object. If a point between the tongs is within the required raduis of the contact point and the tongs are closed to less than that radius, then the object has been gripped. Objects of arbitrary shape can be grasped by using a series of spherical touching conditions. The object is approximated by a group of contact points with different radii. Difficulties arise in determining if the tongs are sufficeintly closed if there is a possibility of more than one contact point being within the tongs. Figure 6 shows a rectangular block approximated by two spheres. If it is grasped such that the tongs close in along the y-axis, the object can be grasped if the center of the tongs are anywhere in the shaded region. However, the tongs would have to be closed to the radius of one sphere or half the width of the block. The block could be approximated by one sphere (figure 7), which would result in space which is not part of the block being an acceptable place to grip the object and the tongs would not have to be fully closed when the block is gripped along its small side.

Nonsymmetric touching conditions allow for more accurate and realistic grasping of objects. However, they require the ability to translate the touching conditions and the position of the tongs to the same reference frame. Since the tongs are described by points and lines, they are most easily transformed into the object reference. This is done

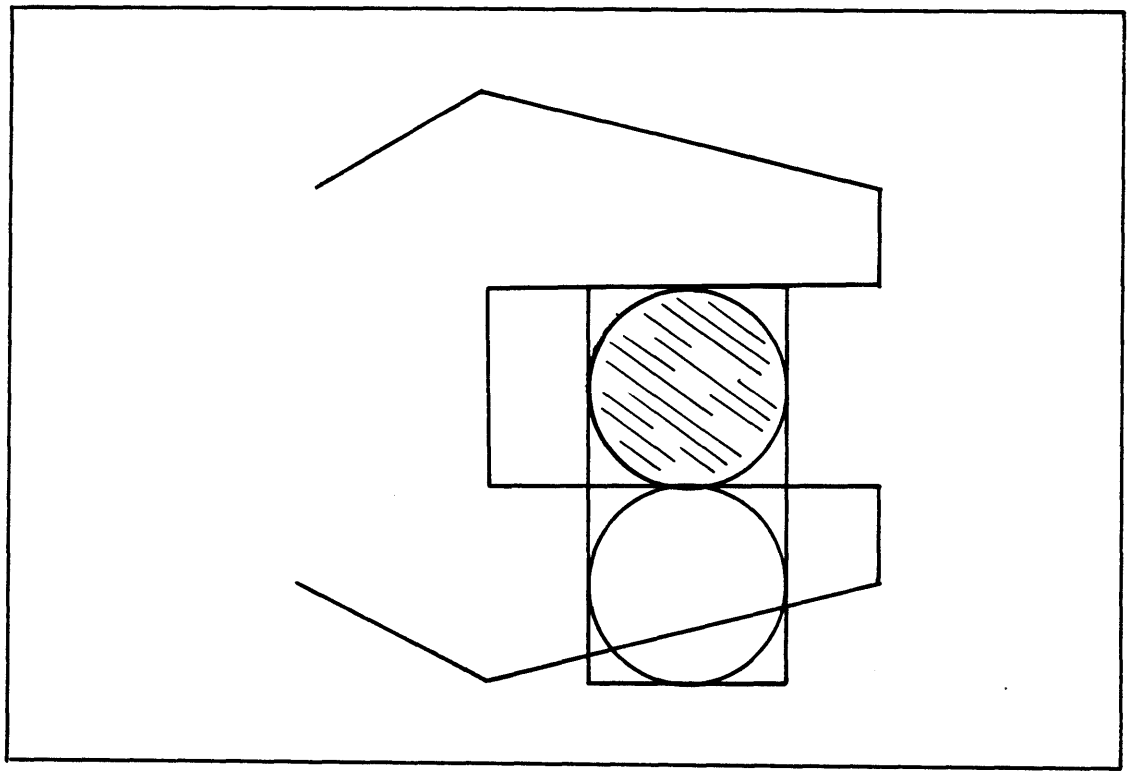

Figure 6 Multiple Spherical Touching Conditions

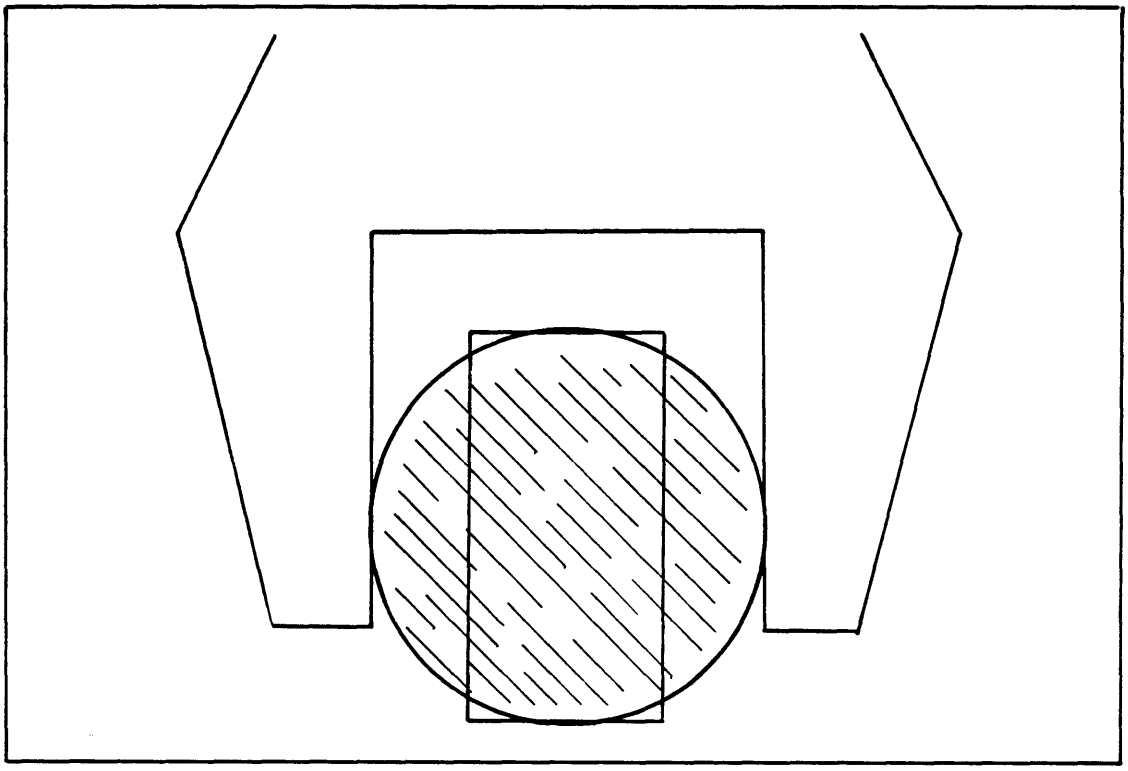

Figure 7 Multiple Spherical Touching Conditions

by premultiplying the points in the tongs by the inverse of the object transformation. Once the tongs and object are in the same reference frame, appropriate touching conditions are needed. Two requirements are placed on touching conditions. First it must be possable to determine if a point satisfies them. Secondly, given a unit direction vector, it must be possable to determine the scaling factor needed to make the scaled direction vector reach the surface of the object. The second condition is needed to determine the closure of the tongs required to grip the object.

The most basic nonsymmetric touching conditions are rectangular conditions. Figure 8 shows two-dimensional rectangular conditions for the rectangle described by length 2A, height 2B and center (Xc,Yc). If the center of the tongs is given by (Xt,Yt) in the object reference frame, then the object is within the tongs if:

$$|Xt-Xc| < A \qquad \text{and} \qquad |Yt-Yc| < B$$

If the object is within the tongs, it is then necessary to check to see if the tongs are sufficiently closed. Let the vector (Xn,Yn) be the normal vector to the face of the jaws. If origin for the normal vector is taken to be the center of the rectangle, it can be scaled to a new vector (Xs,Ys) which intersects the surface of the intersects the rectangle. If this vector were to intersect the line Y=B, then by proportional triangles, its coordinates are found to be:

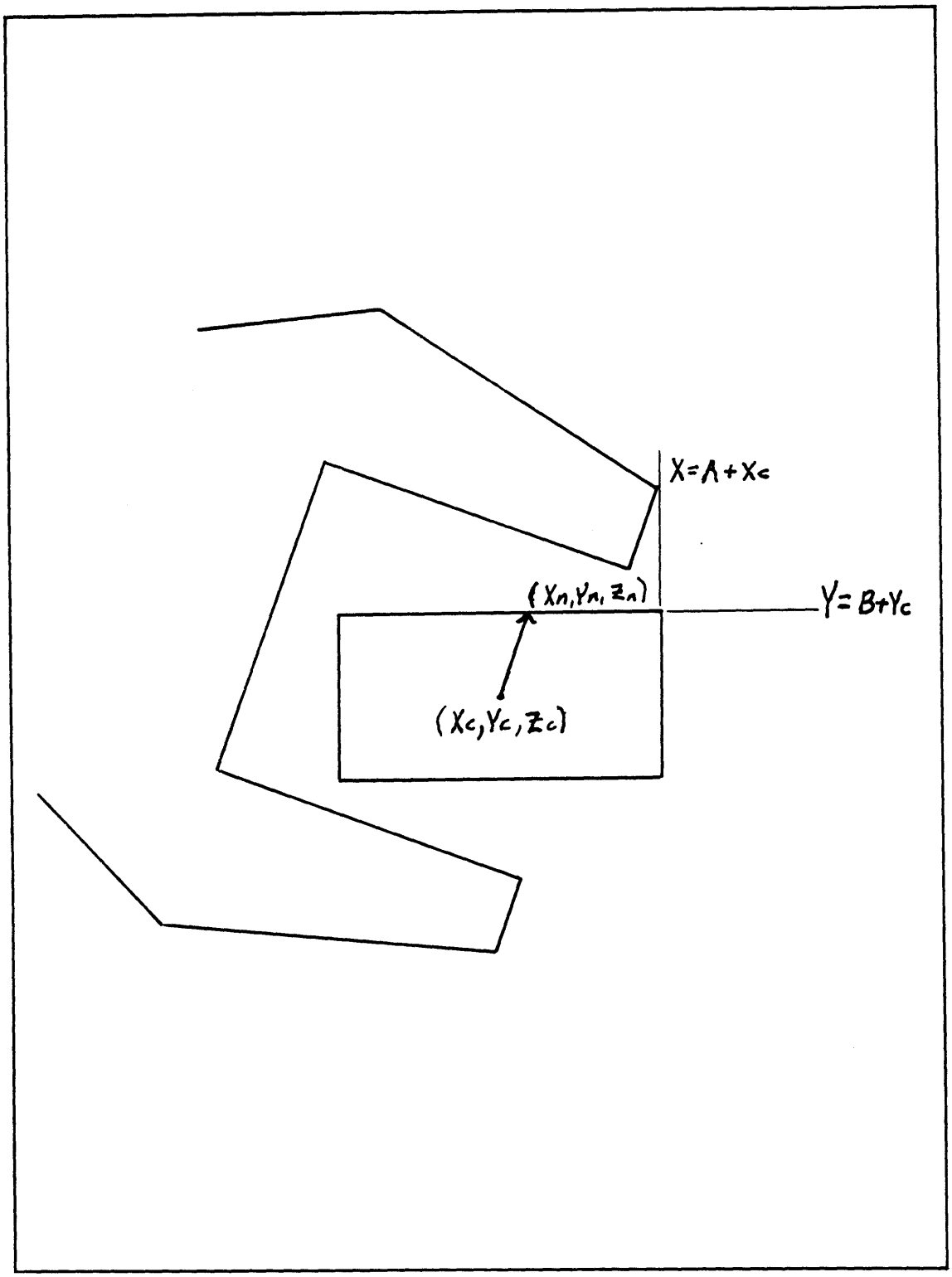

Figure 8 Rectangular Touching Conditions

$$(Xs,Ys)=(A,YnA/Xn)$$

If the vector were to intersect the line X=A, its coordinates are given by:

$$(Xs,Ys)=(XnB/Ys,B)$$

The shorter of the two vectors is the one which stops at the surface of the rectangle and its magnitude corresponds to width of the rectangle as measured in the direction of the normal vector. The tongs are closed enough to grasp the object if the closure D is less the the magnitude of the vector (Xs,Ys).

EQUIPMENT

The major equipment used in this project is shown in figure 9. The simulation was run and controlled from the computer terminal in the center of the picture. The display terminal is setting on the table to the far right. The master manipulator is directly in front of the display terminal. The slave is in the background on the left. The large rack of electronics behind the display terminal is the servo-amplifiers and control interface. The computer is not shown.

PDP11/34 Computer

The simulation was performed on a PDP11/34 running a RSX-11M operating system. This is a multiuser, timesharing system. A PDP11/34 is not a particularly fast computer, however it was sufficient. It would be more desirable to run on

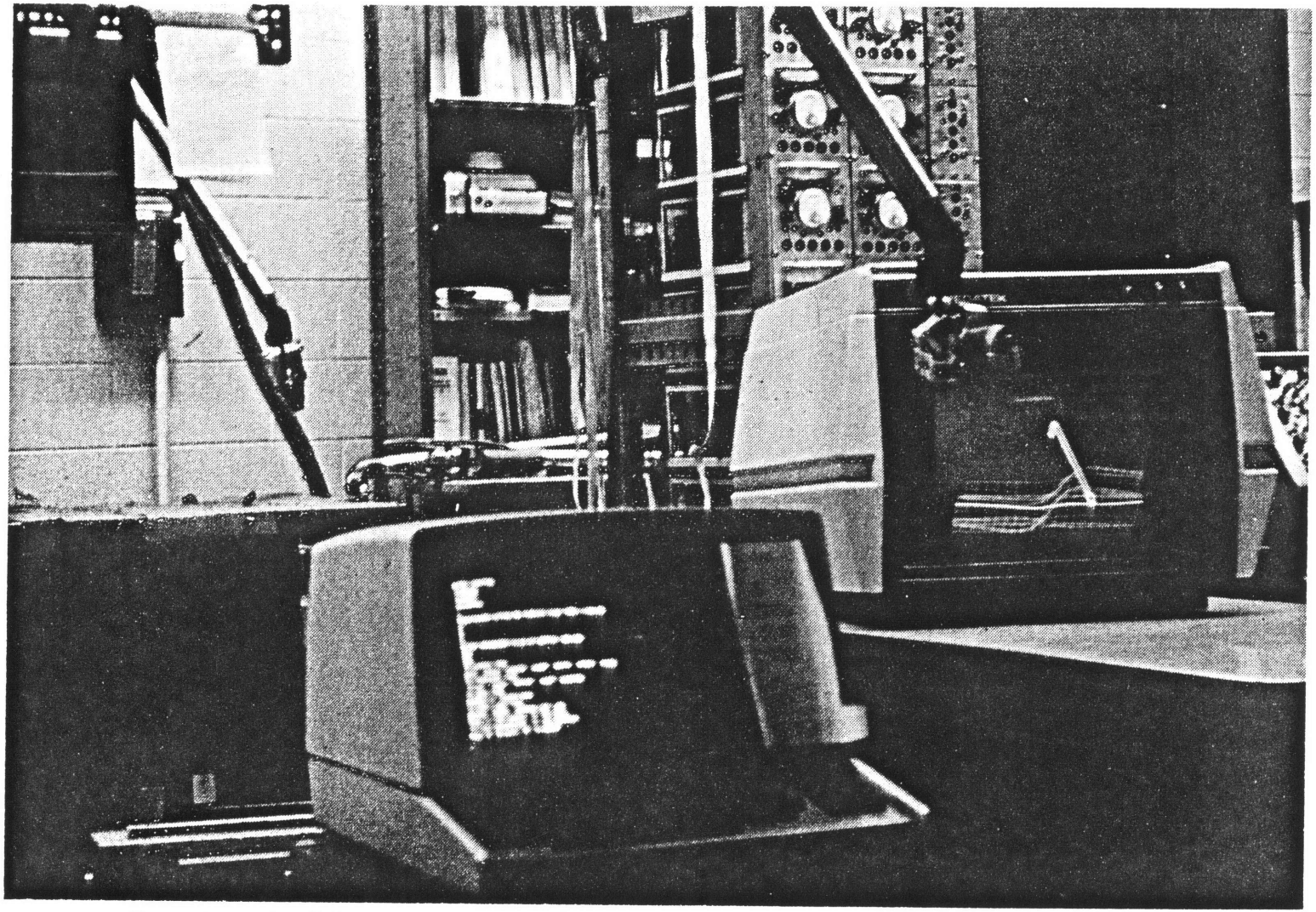

Figure 9 Man-Machine Systems Laboratory

a machine solely dedicated to this task.

E-2 Master-Slave Manipulator

The manipulator was a rebuilt Argonne National Laboritory E2 master-slave manipulator (2). This is a light hot-room style manipulator with seven degrees-of-freedom and full force reflection. It is electronically coupled allowing it to be interfaced to the PDP11/34 through a AN5400 A/D converter using an interface built by K. Tani (7). The control interface allows independent sensing and control of each degree-of-freedom. Figure 10 show the E-2 slave manipulator simulated in this project.

Megatek Display Processor

The graphics were performed on a Megatek 7000 vector graphics terminal with three dimensional hardware rotate and a resolution of 4095 lines. The display processor has the ability to move the beam from its last location to a new location on the screen with the beam either on or off. The beam can also be moved a given displacement from its last position. With appropriate manipulation, text strings can be displayed in a variety of sizes and orientations. A series of commands can be defined to be a subpicture. The commands can then be executed by referencing the subpicture number. This allows a series of strokes which is used many times to be defined only once, then accessed by a single command. This saves time in loading the display processor.

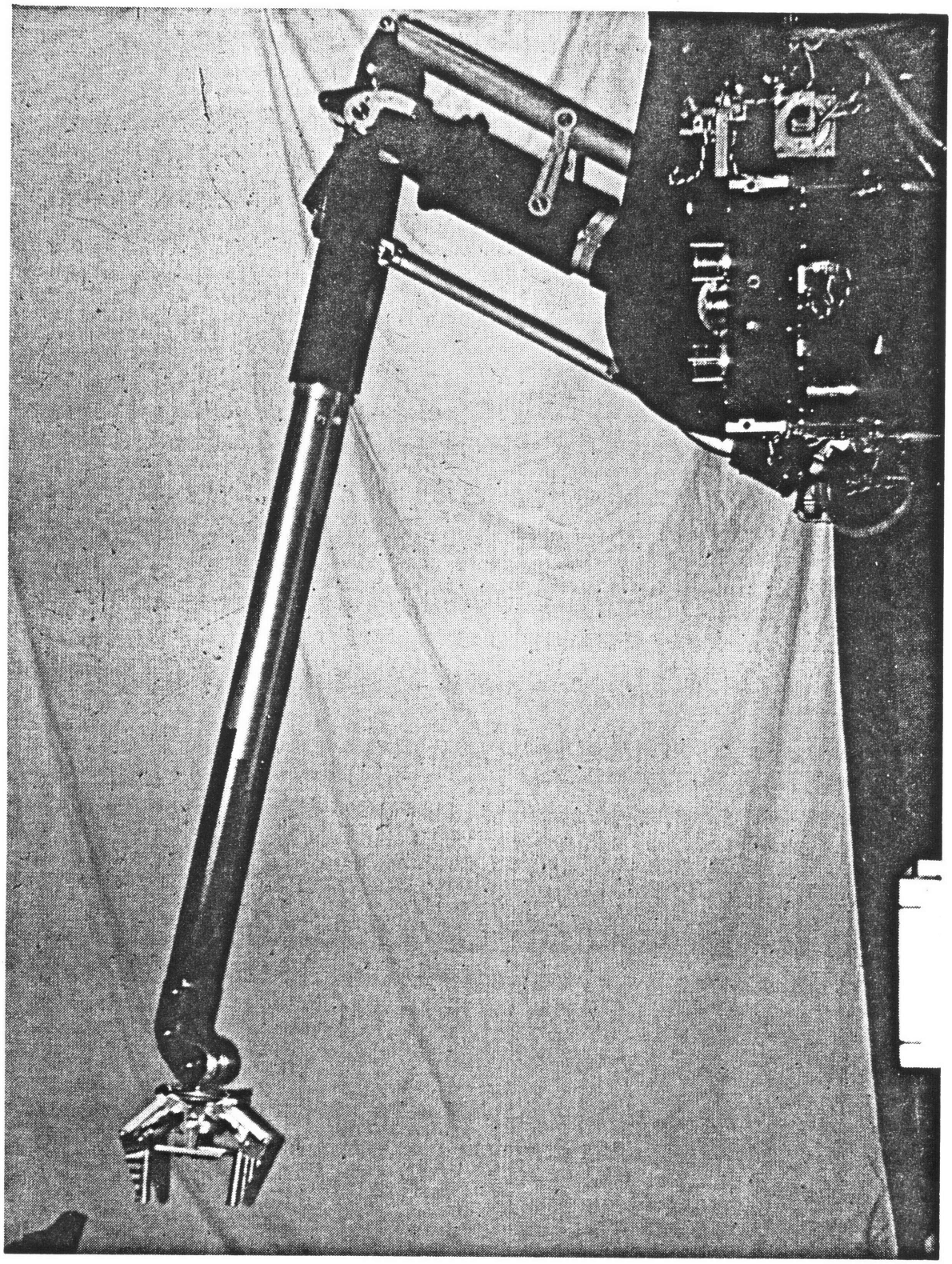

Figure 10 E-2 Slave Manipulator

The hardware rotate is a 4x2 matrix multiplier which is loaded into the display list and affects all succeeding vectors. It is capable of rotation, translation, scaling and clipping at screen boundaries. It is not, however, capable of perspective. The origin for all transformations is assumed to be the center of the screen. Calls to the hardware rotate are not cumulative, but rather each call overrides the preceeding call. Relative vectors are affected by the rotation matrix applied to the first absolute vector preceeding the relative vectors regardless of when the matrix multiplier was called. This caused some problems in moving the sphere which was made up solely of relative strokes.

The above commands are placed in an installed common block in the PDP11/34 and are transferred to the Megatek's display list en masse by a DMA transfer. The Megatek sequentially scans the display list and puts the desired vectors on the display monitor. There are also commands to determine the location of a particular instruction in the the display list and commands to change that location. Any number of commands can be erased from the end of the display list and be replaced by others.

EXPERIMENTAL DESIGN

Programming Considerations

There are two important, competing programming factors in the simulation: speed and size. The RSX11M system

which was operating on our machine restricts program size to 32K words. A 4K address window is required for each installed common block used in a program. One common block was needed to communicate with the A/D and Megatek, and one common block was needed to hold Megatek commands prior to DMA transfer. This means the maximum effective program size was only 24K. Speed was a factor since it was desirable that the program run in real time. It was particularly critical when the program ran simultaniously with H. Kazerooni's dynamics program, which required a cycle time of less than 230 milliseconds to retain stability. Secondly, when computer generated force-feedback was desired, the computer needed to send signals to the manipulator control interface at time intervals on the order of 20 milliseconds.

Most of the graphics program was involved in calculating rotation matrices and manipulating objects. Only a small portion of the program was involved in initialization. This eliminated the possiblity of using overlays or multiple tasks to reduce program size. It was necessary to make compromises between arrays versus explicit variables, and subroutines versus inline program segments. Explicit variables and inline programming are faster, but require more program space. When there are many zero terms in a program section, such as those which occur when preforming multiplications of rotation matrices, explicit programming may be as

short as subroutines and obviously much faster. In general, explicit variables and inline programming were used for speed and ease of interpretation.

The amount of data space required was reduced by placing all arrays in the unused portions of the common block associated with the display processor. All point data for the display was placed in direct access files, eliminating the need to bring it into the program. This also made program size independent of display complexity.

Using the display processor's hardware rotation and subpictures reduced cycle time by a factor of three over the equivalent program without these options. During the initialization segment of the program, each independent element of the display was drawn into a subpicture in its unrotated form. In the main body of the program, the display was created by loading the appropriate transformation matrix into the display processor's hardware rotate, followed by calling the desired subpicture. The use of subpictures relieved the computer from the task of reloading the entire display each program cycle. The hardware rotate eliminated the need to individually calculate the coordinates of each point in the display.

When the manipulator was allowed to move freely under computer control it was critical that the manipulator be controlled at regular time intervals. Since these intervals

were shorter than the cycle time of the display program, control of the manipulator during free motion was performed by a separate program. When the manipulator was touching the simulated surface and force-feedback was being generated, the cycle time was not as critical because the velocity of the manipulator was small. Since in this case the display program needed to calculate the new angular position of the manipulator required to generate force-feedback, the display program controlled the arm while the control program remained dormant. Syncronization between the two programs was achieved by the use of flags passed through an installed common block.

Manipulator and Vehicle Simulation

A design for the manipulator was chosen which simply but accurately represented the shape of the E-2 manipulator. Since no hidden line removal was attempted in order to keep the cycle time of the program down, too many lines would have made the display difficult to view. The tongs which are the most important portion of the display were given the most detail. The forearm and shoulder had a square, rather than circular cross-section. Cylinders are difficult to display correctly unless complex algorithms are used to determine the location of their edges. The square cross-section also gave a better perception of rotation of than a circular cross-section.

Two types of objects and touching conditions were implemented. The first object implemented was a sphere using spherical touching conditions. When this proved successful, a rectangular peg was installed. The peg used two sets of rectangular touching conditions. One for the main body of the peg and one for the stem.

Various dynamic and static properties could be inputted for the objects allowing a simulated environment to be built up within the computer. Gravity, drag, elasticity of collision, conservation of momentum and the hardness of the objects were all parameters which could be entered into the simulation. The manipulator and objects were enclosed within a rectangular room. When a moving object collided with a wall of the room, it rebounded off the wall. Aside from demonstrating conservation of momentum, the main purpose of the walls was to keep the objects within the reach of the manipulator.

Two applications of force-feedback were introduced. When an object was gripped by the manipulator, force-feedback was sent to keep the tongs open to the width of the objects. The resulting sensation was that of an actual object within the tongs. The second application involved applying force-feedback to the full manipulator. A three-dimensional surface was defined. The surface was assumed to be relatively flat eliminating the need to calculate the surface nor-

mal, instead the surface normal was assumed to be vertical. Different hardnesses were assigned to various locations on the surface. Force-feedback was implemented so that the surface could be felt when it was touched by the manipulator. As visual feedback, a gridwork approximation of the surface was displayed on the graphics terminal. To aid the operator in perceiving depth, the contour directly below the manipulator was displayed in darker linework. As the manipulator penetrated the surface, the contour deflected. The extent of the deflection was dependent on the stiffness of the surface and depth of penetration. If the surface was soft, deflection only occurred in the neighborhood of the penetration. If the surface was stiff, the whole surface deflected.

The simulation could be displayed from any viewpoint. The viewpoint could be either stationary or moving. It has been suggested that a moving viewpoint might be useful in giving the operator a better perception of the three-dimensional nature of the environment (6). The display could be scrolled and zoomed so that any portion of the display could be observed in detail. The viewpoint could be controlled from the keyboard without interrupting the program by use of a request for IO (QIO).

The vehicle simulation was capable of the same functions as the manipulator simulation. The manipulator was mounted on a vehicle which was capable of six degree-of-

freedom motion. The position of the vehicle was inputted through an installed common block, which allowed the vehicle to be controlled by a secondary program.

Depth Indicators

A major difficulty associated with a vidio terminal is the lack of depth perception. Normally, when one is looking at a three-dimensional scene, one's brain generates the peception of three-dimensions from differences in parallax between the scene veiwed through the right and left eyes. The differences in parallax are associated with the distance between the two eyes and the distance between objects and the eyes. When a scene is displayed on a vidio terminal, all points are at the same distance from the eyes and the parallax information is lost.

When the simulation was first developed, depth information was transmitted using the traditional orthographic projections. This approach appeared to cause some coordination problems and the operator seemed to become confused over which view was the front and which was the side. Experienced manipulator operators often rely heavily on shadows for depth cues. The second depth indicator was a shadow with the source of illumination directly overhead. The shadow was cast on a imaginary horizantal floor 50 inches below the manipulator's shoulder. Walls displayed on the screen were helpful in orienting the shadow. Both of these indicators

require extensive prior knowledge of the environment. The third depth indicator was a proximity indicator. The indicator showed the absolute distance between the tongs and the object as a line on the display terminal. The display was designed such that the length of the line was the same scale as the display when the object was within 24 inches of the tongs. The indicator was ten times less sensitive at longer ranges. The proximity detector could be implemented by placing a sonar device on the tongs of a manipulator.

Experimental Measurement of Operator Performance

The three depth indicators plus a control were tested on five subjects. Two types of tasks were designed. The first involved reaching out and grabbing a two inch, stationary sphere. The time required for a subject to grasp the sphere after the display was flashed on the screen was recorded. In the second experiment, the task was the same however the sphere was moving. The path of the sphere was an orbit about the surface of an ellipsoid described by:

$$X=Xc+10\sin(t/3)\cos(t/12)$$
$$Y=Yc+5\sin(t/3)\sin(t/12)$$
$$Z=Zc+7\cos(t/3)$$

where (Xc,Yc,Zc) was a vector describing the center of the ellipsoid, t was the time in seconds and distances were measured in inches. It was hoped that the moving task would bring out any coordination problems associated with the

depth indicators.

All three display types were shown at the same scale. The shadow was displayed along with reference walls at an azimuth and zenith of 15 degrees (figure 11). The second display type was a front view with a proximity indicator displayed vertically above the manipulator (figure 12). The third display type was a front and a side orthographic projection shown side by side (figure 13). The control was the proximity indicator with no view of the manipulator (figure 14).

The subjects were run through a series of display types with both moving and stationary objects. The positions of the objects had to be selected such that the objects remained within the reach of the manipulator and did not coincide with any real object which would obstruct the manipulator. Display types were mixed because subjects tended to become bored with repetitions of the same display type. The order of the display types was kept constant so that the subjects knew which display type to expect next. The positions of the sphere were arranged such that the average distance between successive positions was the same for each display type.

RESULTS AND CONCLUSIONS

Master-Slave Simulation

As the simulation developed, it became increasingly

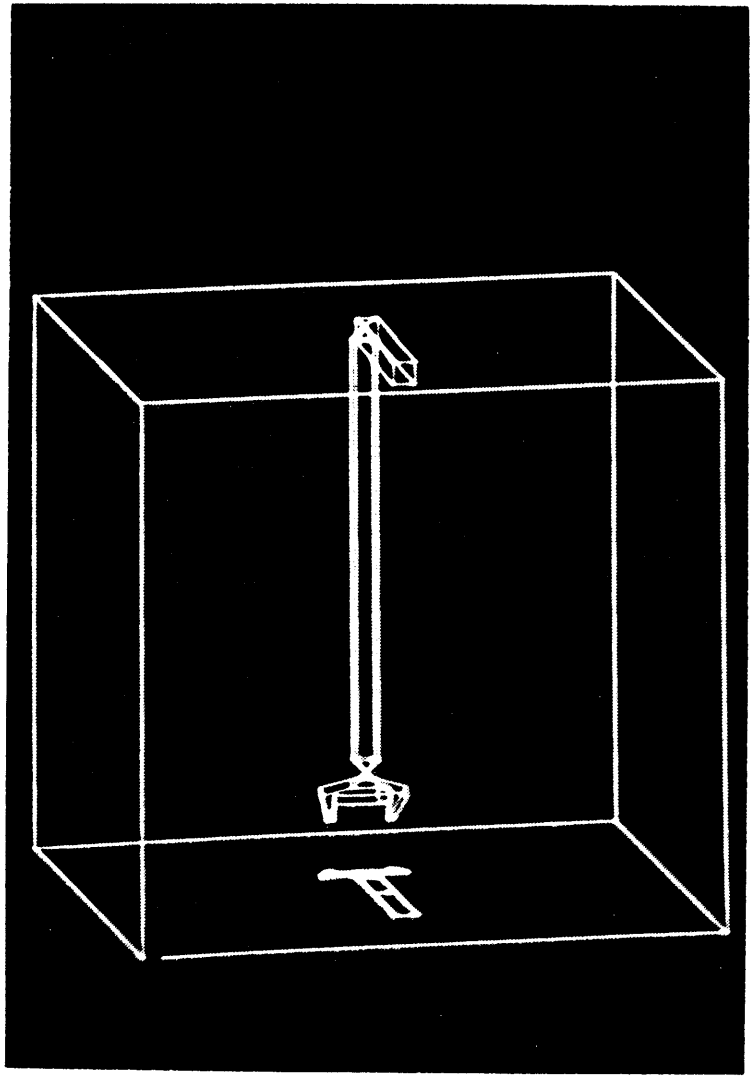

Figure 11 Display Type 1
- Shadow with Walls

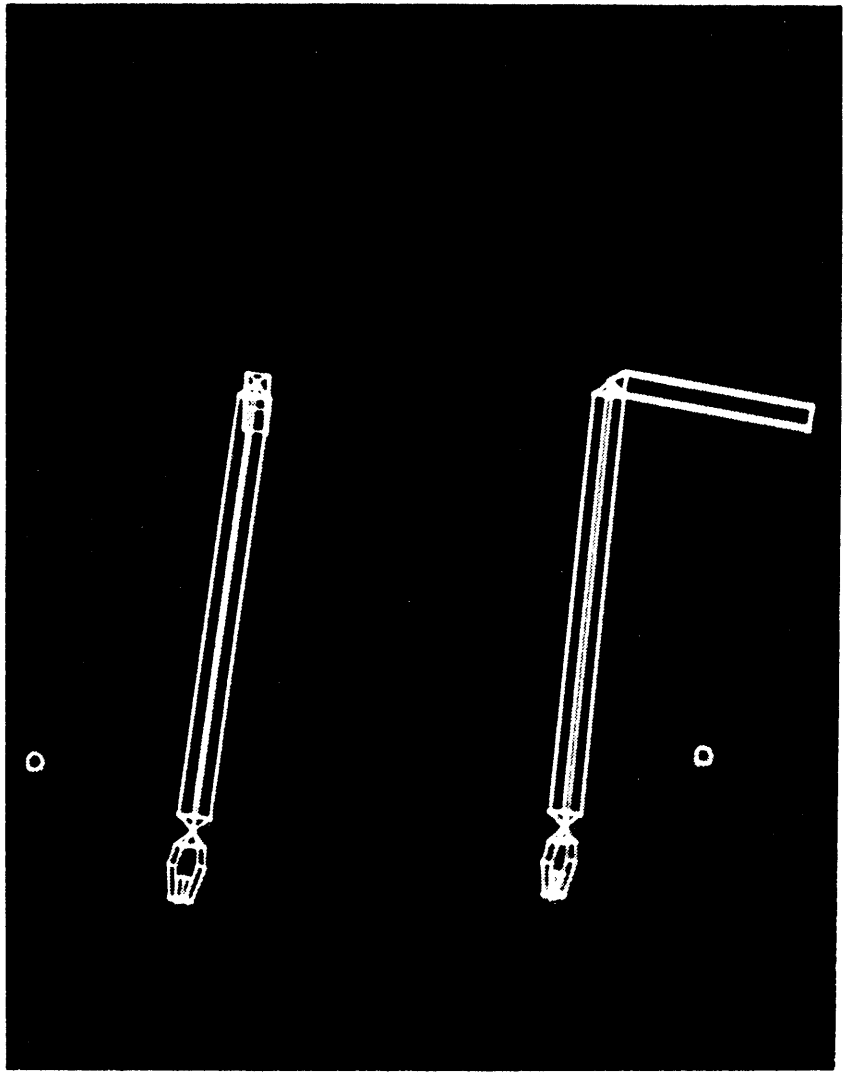

Figure 12 Display Type 2
- Front and Side Isometrics

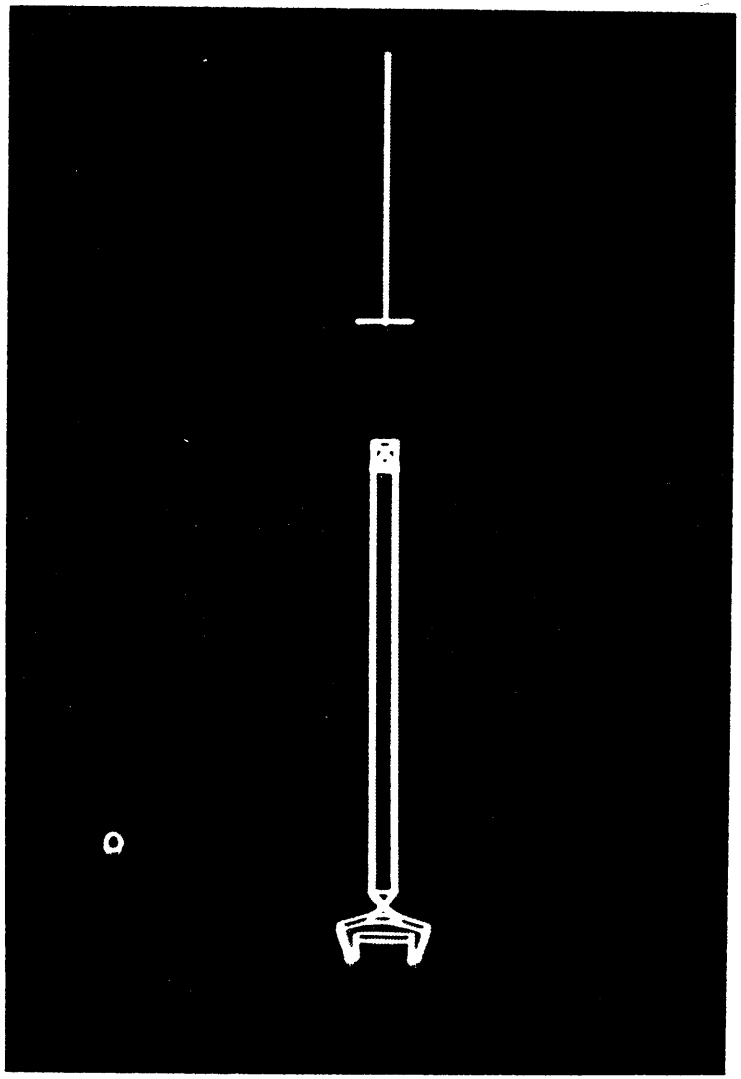

Figure 13 Display Type 3
- Front View with Proximity Indicator

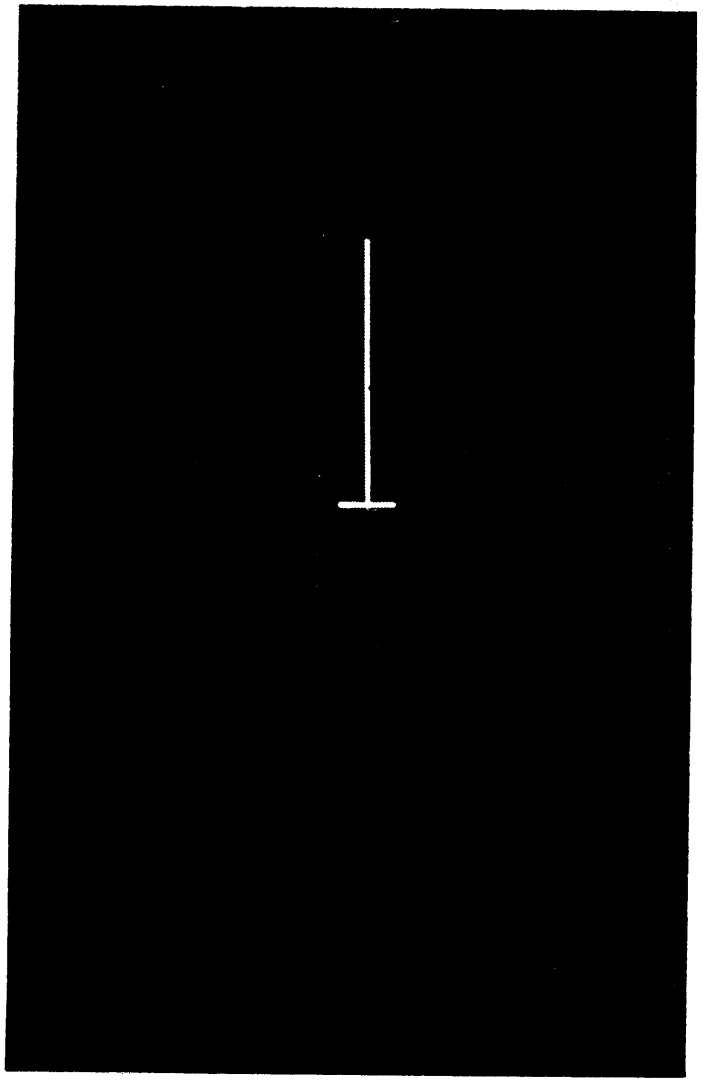

Figure 14 Display Type 4
- Proximity Indicator Alone

apparrent that that the cycle time of the program was an important issue. By using the techniques mentioned under Programming Considerations, it was possable to keep the cycle time of the program to 60-100 milliseconds. This seems to be quite tolerable.

Figures 15 and 16 show the simulated manipulator and its shadow from various viewpoints. The sphere and the object can also be seen in these illustrations. The walls are needed as a reference for interpreting the shadow. The shadow is cast on the floor of the cube. Figure 17 shows the tongs grasping a rectangular peg from several different orientations. It is important to note that the closure of the jaws is dependent on the orientation of the object. The object is centerred between the jaws when it is gripped.

Figures 18 and 19 show the simulated surface. The dark contour is the profile of the surface directly below the manipulator. Figure 19 shows the surface being deflected by pressure from the simulated manipulator. Figures 20 and 21 show the differences in deflection of a soft surface (figure 20) and a hard surface (figure 21). The deflection of a soft surface is much more localized when compared to the deflection of a hard surface.

The submarine simulation was capable of the same functions as the manipulator simulation. It was also capable of motion in six degrees-of-freedom under control of a se-

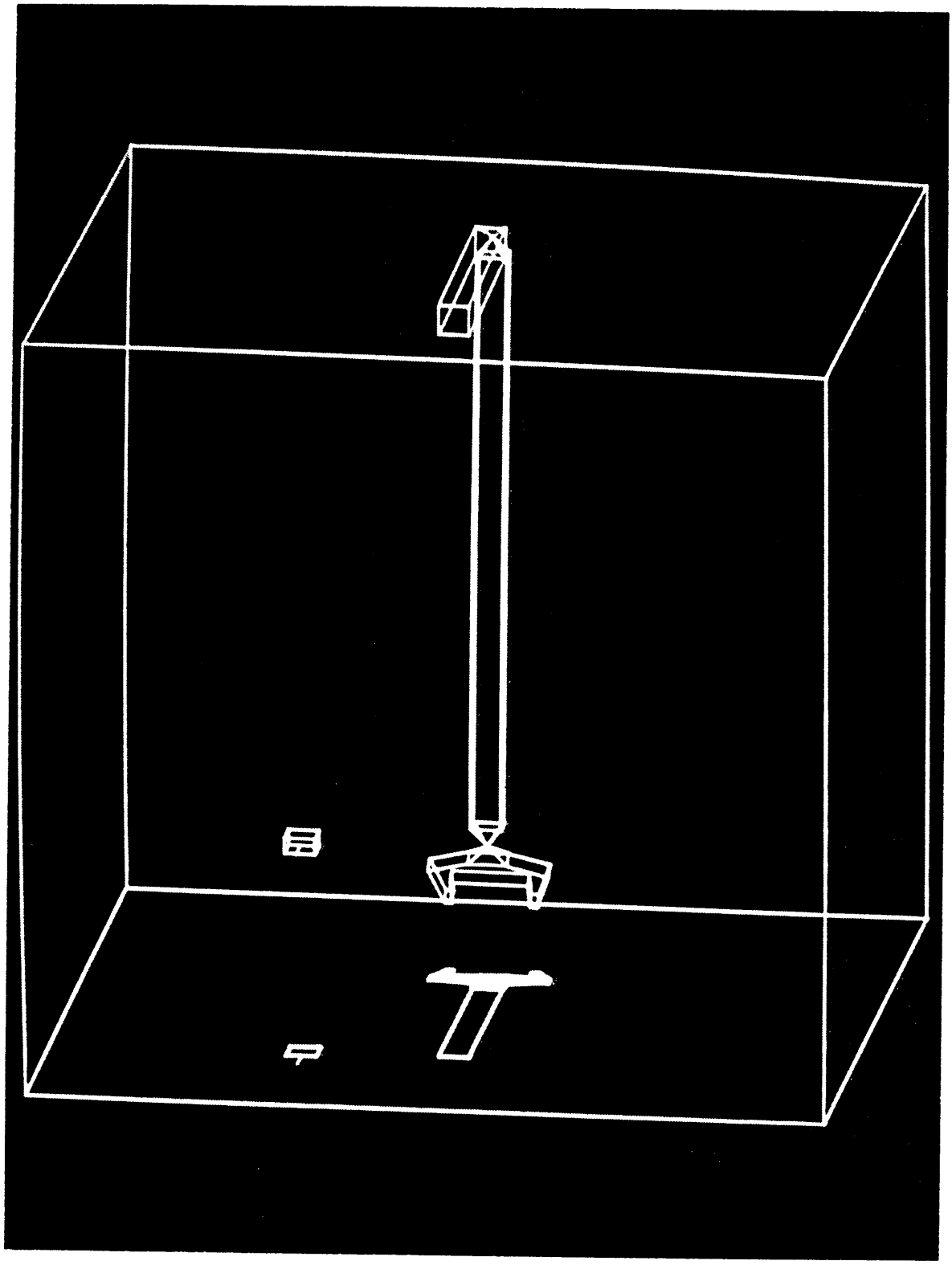

Figure 15 Manipulator Simulation with Shadows

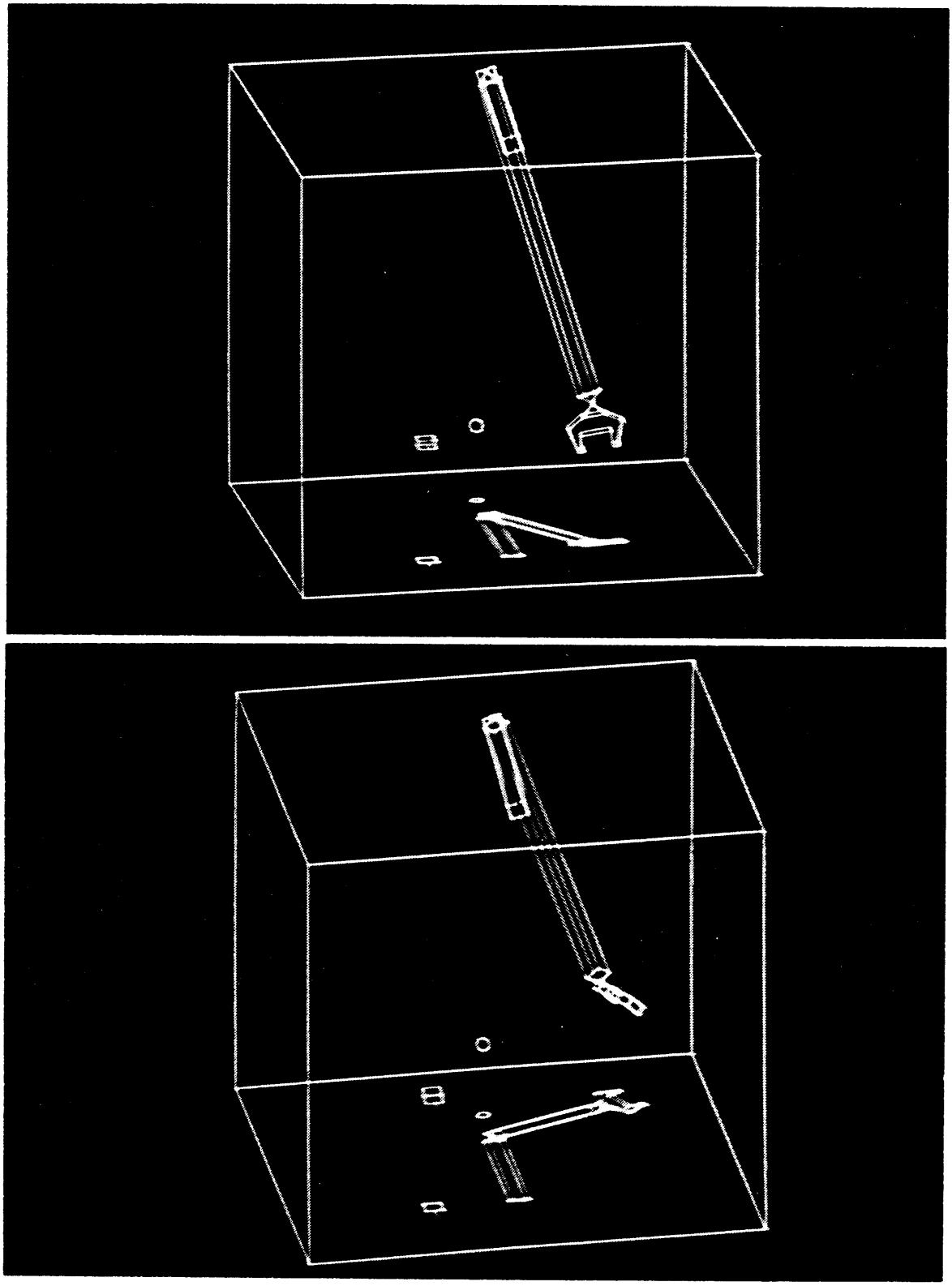

Figure 16 Manipulator Simulation with Shadows

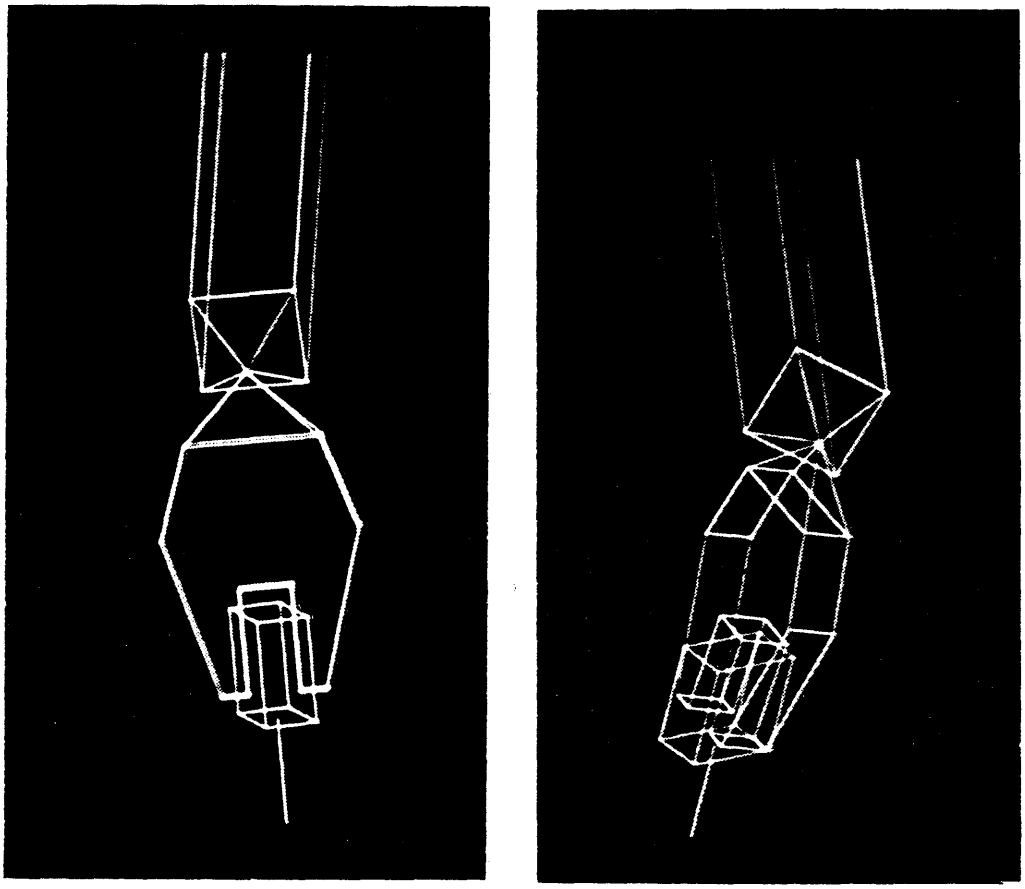

Figure 17 Detail of Tongs Gripping a Rectangular Peg

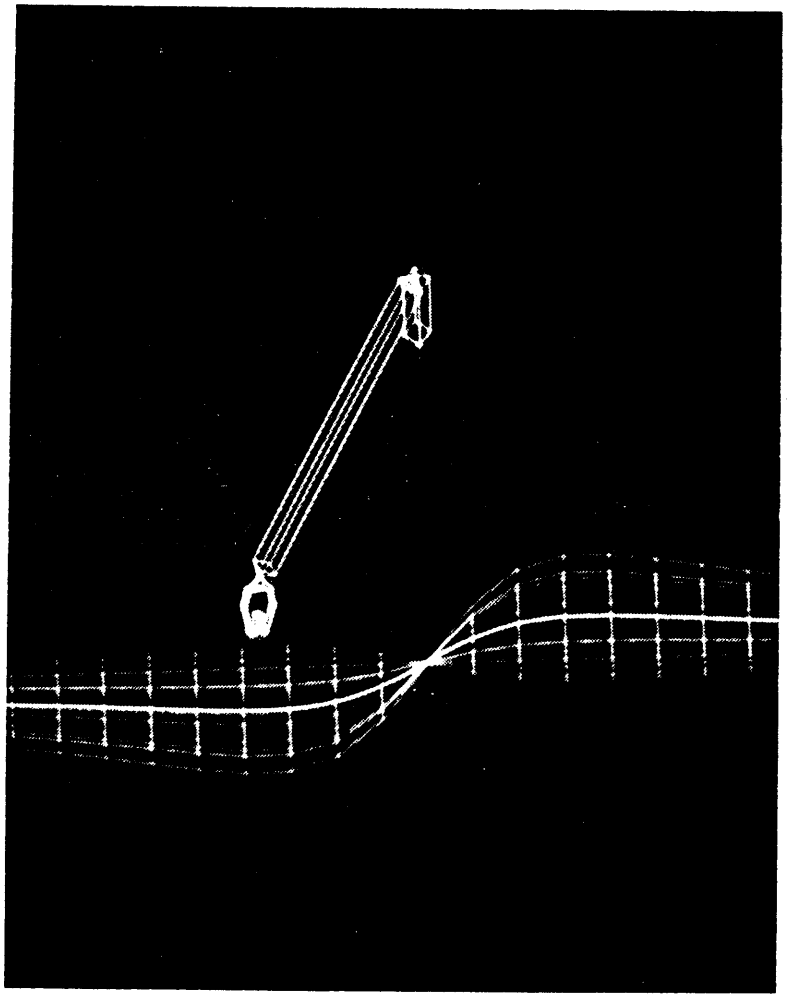

Figure 18 Simulated Surface

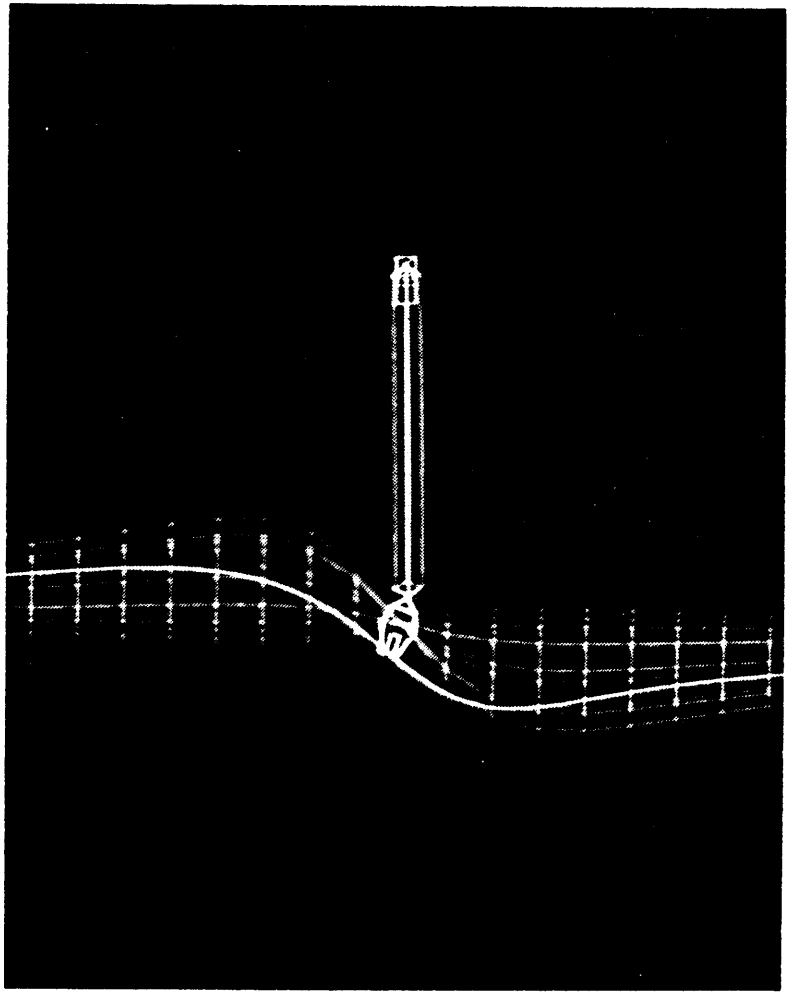

Figure 19 Deflection of Simulated Surface

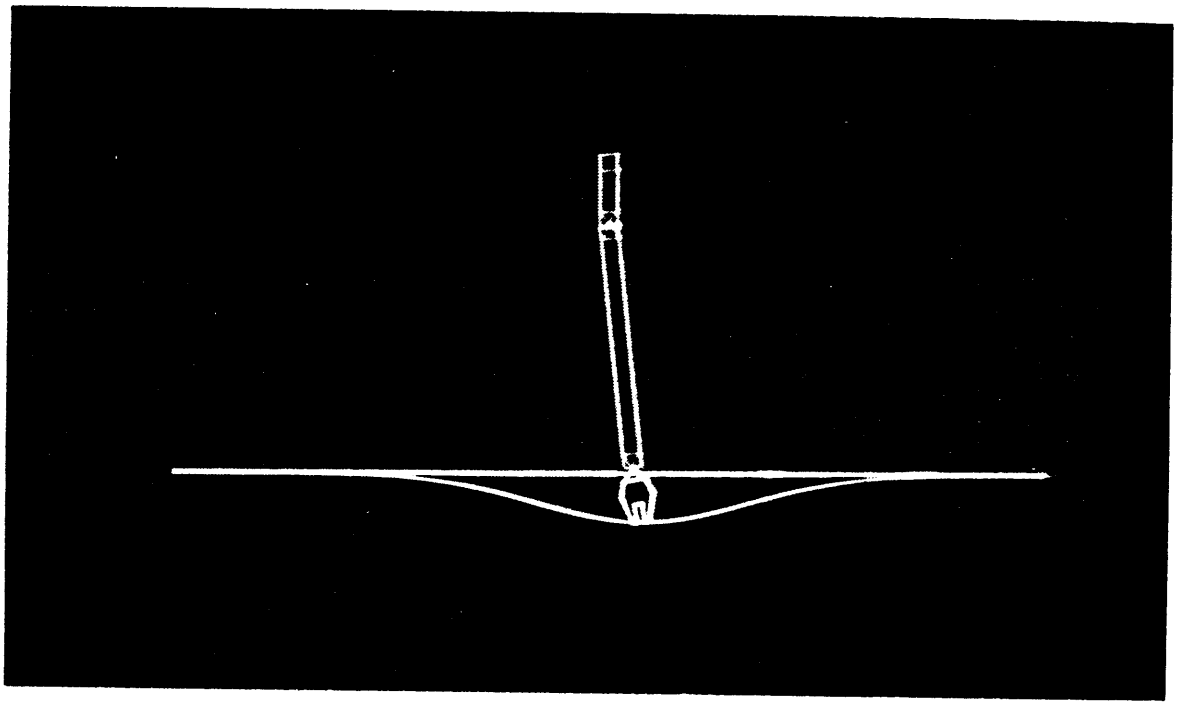

Figure 20 Deflection of a Soft Surface

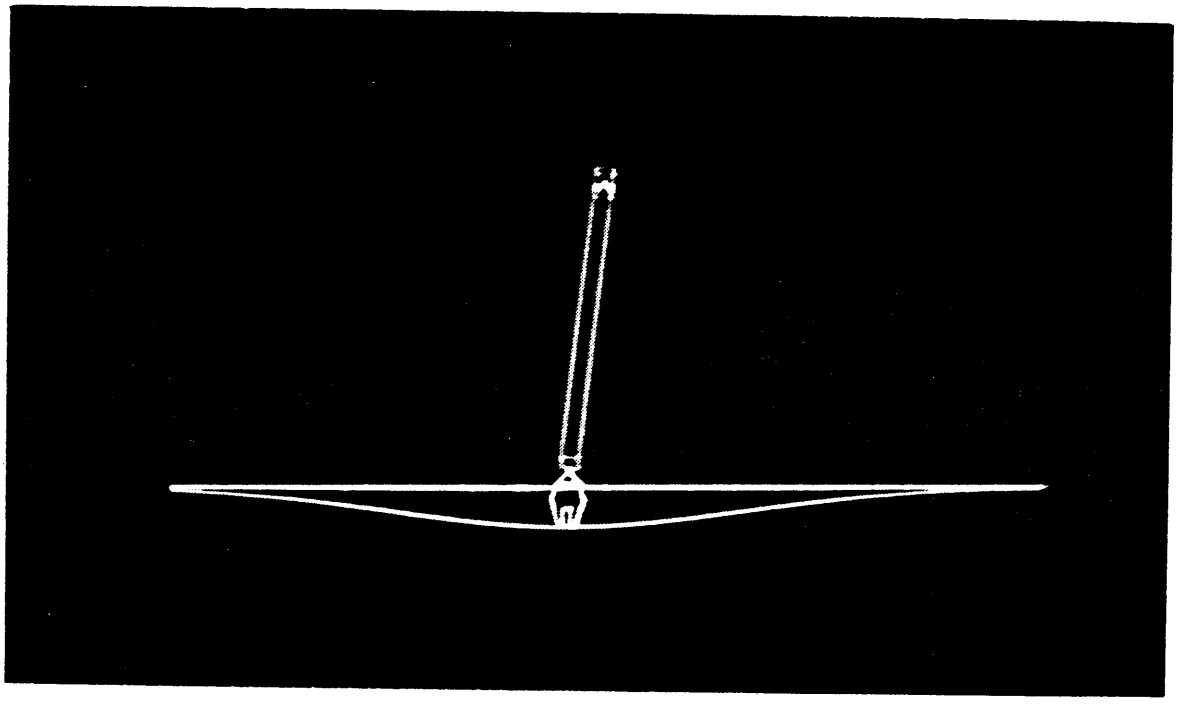

Figure 21 Deflection of a Hard Surface

condary program. Figure 22 shows the details of the simulated submarine.

Modifications of the structure of both the manipulator and the submarine could be made easily by changing the point-connectivity data bases. This allowed the displays to be easily adapted to represent the submarine or manipulator being simulated at the time.

Simulated Force-Feedback

One feature of the simulation which was quite successful but was not formally evaluated was simulated force feedback. Force-feedback generated when grasping an object gave the simulation a strong feeling of reality. It could be used at two levels. The first level was conformational. On this level, the force-feedback was either on 100 percent or it was off. This was highly useful when grasping an object because it gave tactile conformation of the object being in the tongs. If the object happened to slip from the tongs, the loss of tactile feedback gave an immediate indication. Though the operator could tell by looking at the graphic display that the object was no longer in the tongs, if he was preoccupied with other tasks, he might not notice for several seconds. The change in tactile feedback was impossable to ignore. The second level was quantitative. On this level, force-feedback could contain information about hardness or weight etc.. Surfaces varying between very rigid

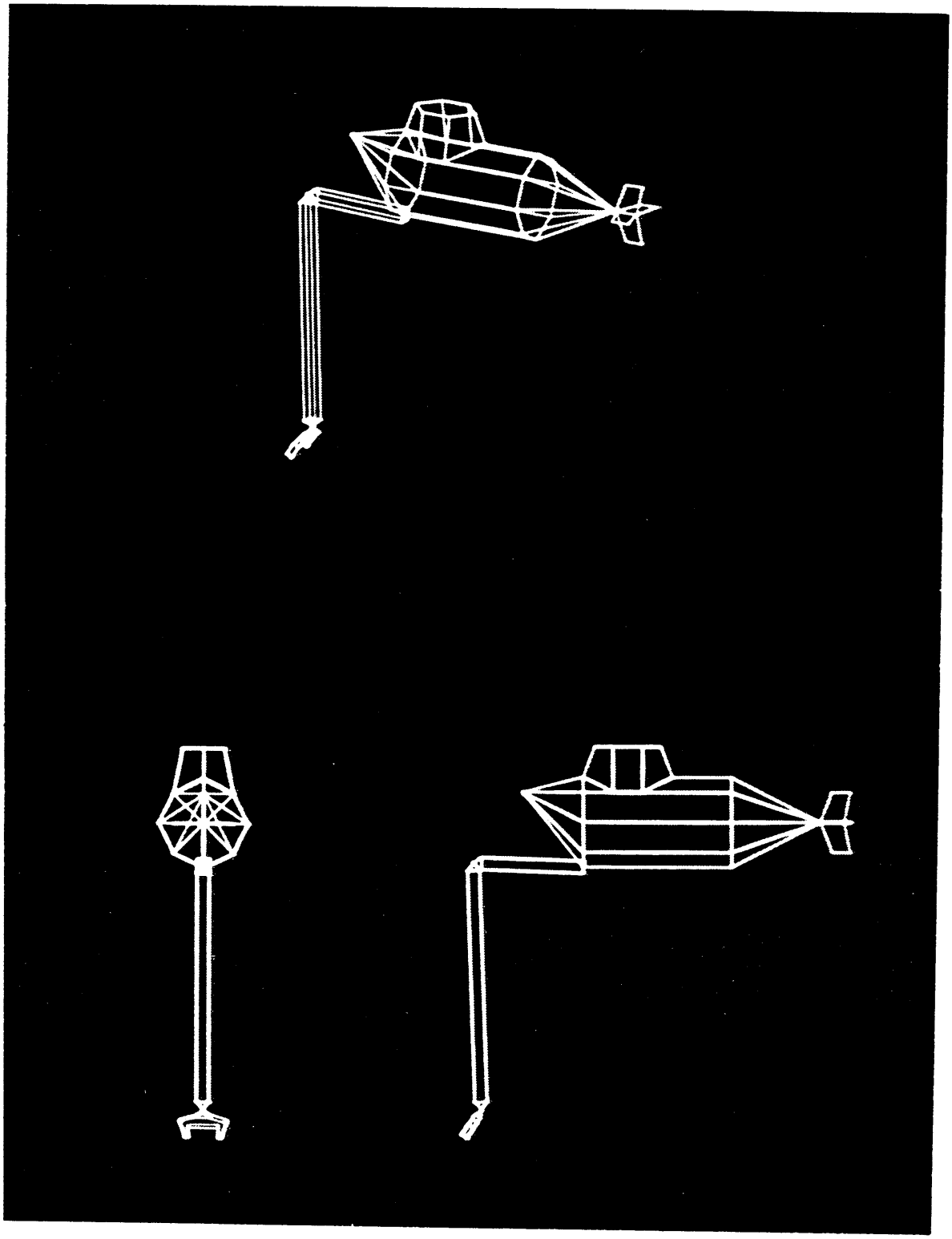

Figure 22 Submarine Simulation

and the consistancy of foam rubber were simulated. The greatest difficult with simulating a variety of types of surfaces came from the cycle time of the simulation. As a surface becomes softer, deeper penetration was possable for a given applied force. This larger range of motion implied that the manipulator was required to be moved larger distances each program cycle. The simulation program had difficultly updating the manipulator quickly enough. The result was that the manipulator moved in discrete steps when feeling a soft surface. Despite this difficulty, it was possable to simulate a large range of hardnesses. Although the manipulator response was not perfect for soft surfaces, its response was tolerable. Overall the simulation of force-feedback seems quite promising.

Depth Indicators

Each of four subjects preformed 80 to 90 repetitions of each the four display types and two tasks. Subjects 1 and 2 were experienced with both the E-2 manipulator and the graphic simulation. Subject 4 was experenced with the manipulator and subject 3 had no prior experience. All subjects learned to use the displays in one to two hours. Learning curves were recorded for each subject (Appendix I). When these curves showed no significant learning, the subjects performed an additional 32 repetitions of each task. The data from these final trials was analysised. The overall re-

sults are summarized in figure 23. Complete results are found in appendix I. The proximity indicator with no additional display clearly gave very poor results. In order that this would not overwelm the other three display types, the proximity indicator was left out of the statistical analysis. A three-way analysis-of-varients was performed on the remaining three display types (Appendix I). The results showed significant differences between subjects, display types and the two tasks. There were significant interactions subjects and display types, and between subjects and tasks. There were no significant effects of tasks on display type.

The front and side orthographic projections showed the best performance on both tasks. The subjects felt that it presented only slight coordination problems. Three of the four subjects preferred this display because they felt it gave the clearest detail.

The shadow showed the next best times. All subjects felt that the shadow gave them the best perception of the position of objects in the environment. The main reason they found it difficult to use was that the shadow of the hand tended to obscure the shadow of the object when the two were in close proximity. If the subject was very close to the object and needed to know which way to move, he had to back away from the object to the get a clear picture of the shadows. Although the results were not statisticly signi-

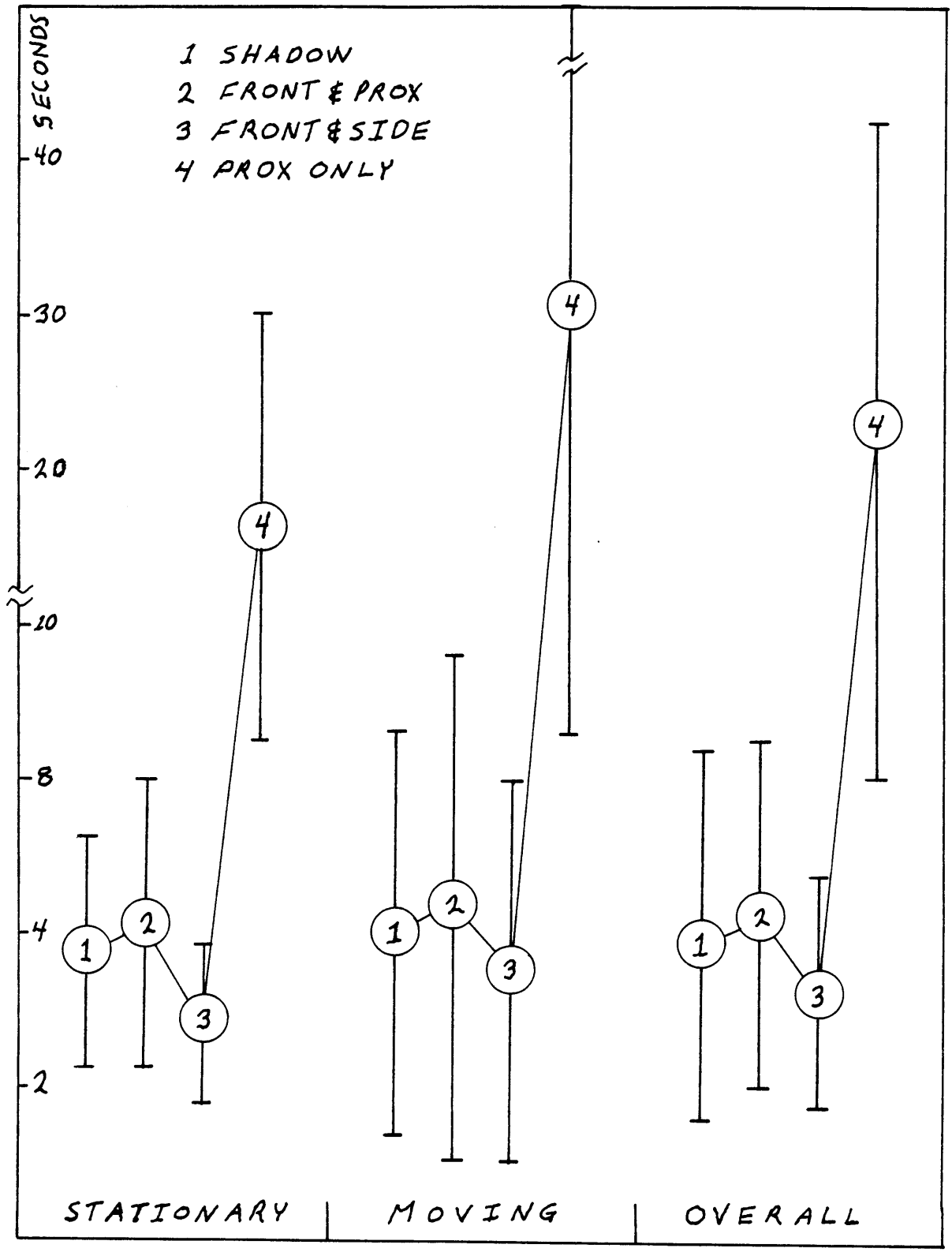

Figure 23 Evaluation of Depth Indicators

ficant, the shadow was less affected by the motion of the object than the front and side views. Larger or faster motions may show more significant differences. The shadow might be improved by reducing the intensity of the linework in the shadow of the manipulator or reducing the number of lines. The shadow may show better performance on a larger object.

One subject preferred the front view with the proximity indicator because of its simplicity. Times using the proximity indicator were close to those of the shadow. In order to obtain depth information from the proximity indicator the operator had to move the manipulator and watch the response of the indicator. This seemed to cause little trouble to the operator though he would occasionally end up searching in the wrong direction before realizing his mistake. Although the proximity indicator could be easily implemented, it gives only limited depth information and probably would not be suitable for a complicated task.

The proximity indicator by itself was much more difficult to use than any of the other displays. Subjects did learn a method of moving the manipulator in order to determine the proper direction of motion. The proximity indicator was extremely difficult to use on a moving object. The object would tend to change direction of motion just as the subject was getting near to the object.

Although there were differences between the depth indicators, these differences were small compared to the time required to perform the task. It is difficult to say with certainty which is best since each has its good and bad points. The best depth indicator would appear to be a combination of the front and side views with the shadow. The shadow could be used to obtain an overall perception of the environment while the front and side views could be used for detail work. With a computer graphic simulation, allowing the operator to select the best view would be a simple matter.

Conclusions and Recommendations

The manipulator simulation has proven to be a highly flexible and valuable tool. It is presently being adapted to perform several types of function. H. Kazerooni is using the vehicle simulation to test and evaluate different types of control systems for small submersibles. One of the projects is to design a fuel-optimizing, bottom-following routine. The vehicle simulation allows the operator to know the position and orientation of the vehicle in real time instead of needing to interpret many pages of computer output. The visual feedback also allows the vehicle to be driven under manual control. It is hoped that manipulator-vehicle interactions can be integrated into the simulation in the near future.

D. Fyler is examaning methods of defining the environment by feeling about with a manipulator and recording any contact with an object using a variation of this program. The data obtained is then analyzed and a polyhedron is fitted to the data points. This could prove to be highly useful in situations of poor visablity. B. Wood is performing similar tests in which a predefined object is inserted into the display by feeling for reference points which orient the object.

The concept of a manipulator interacting with a simulated environment has proved to be very interesting. It may be fruitful to improve the reality of the simulated environment. Areas of improvement might include addition of the conservation of angular momentum of the simulated objects. At the present time the only interaction between the object and the manipulator occurs when the object is gripped. The simulation could be improved by considering collisions between the object and other portions of the manipulator. This could allow the object to be pushed by the manipulator without being gripped. The use of force-feedback could be expanded to allow the object to feel heavy when picked up, or to allow the operator to feel a reaction force when the manipulator was struck by a moving object.

REFERENCES

1. Brooks, Thurston L.,"Superman: A System for Supervisory Manipulation and the Study of Human/Computer Interactions", Masters Thesis, Department of Mechanical Engineering, M.I.T., May 1979, MIT Sea Grant Report MITSG 79-20.

2. Crandall, Stephen H., Karnopp, Dean C., Kurtz Jr., Edward F. and Pridmore-Brown, David C., Dynamics of Mechanical and Electrical Systems, McGraw-Hill, 1968.

3. Deghuee, Bradley J.,"Operator-Ajustable Frame Rate, Resolution, and Gray Scale Tradeoff in Fixed-Bandwidth Remote Manipulator Control", Masters Thesis, Department of Mechanical Engineering, M.I.T., Sept. 1980.

4. Faux, I.D. and Pratt, M.J., Computational Geometry for Design and Manufacture, Ellis Horwood, 1979.

5. Meyfarth, Philip F., 2.157 Computer-Aided Design class notes, M.I.T., 1979.

6. Sheridan, Thomas B., M.I.T., personal communication,1981.

7. Tani, Kazuo, "Supervisory Control of Remote Manipulation with Compensation for Moving Target", Man-Machine Systems Laboratory Report, M.I.T.,1980.

8. Whitney, D.E., "Resolved Motion Rate Control of Manipulators and Human Prostheses", IEEE Transactions on Man-Machine Systems, Vol. MMS-10, No. 2, June 1969.

APPENDIX I

DEPTH INDICATOR EXPERIMENT STATISTICS

Four subjects performmed the experiments testing the various depth indicators. Their learning curves are shown in figures 24, 25, 26 and 27. When these learning curves became relatively flat, indicating that the subjects had learned to use the displays, each subject performmed 32 additional repetitions of each display type. The means ($\bar{X}$) and the standard deviations (S) for these tests are compiled in tables 1-5.

TABLE 1 AVERAGE TIME TO LOCATE OBJECT (seconds)
SUBJECT #1

TASK / DISPLAY	stationary $\bar{X}$	stationary S	moving $\bar{X}$	moving S	both tasks $\bar{X}$	both tasks S
shadow	2.87	1.00	3.06	1.35	2.97	1.19
front & prox	3.83	1.76	3.95	1.57	3.89	1.67
front & side	3.08	1.14	3.60	1.30	3.34	1.28
prox only	13.54	10.74	32.45	24.40	23.00	21.03
all displays	5.83	7.08	10.77	17.55	8.30	13.59

TABLE 2 AVERAGE TIME TO LOCATE OBJECT (seconds)
SUBJECT #2

TASK / DISPLAY	stationary $\bar{X}$	stationary S	moving $\bar{X}$	moving S	both tasks $\bar{X}$	both tasks S
shadow	3.89	.95	3.09	1.47	3.24	1.30
front & prox	3.77	1.07	3.54	1.26	3.66	1.17
front & side	2.34	.68	2.41	.76	2.37	.72
prox only	12.55	8.16	23.65	18.77	18.10	15.50
all displays	5.50	5.79	8.12	13.00	6.84	18.17

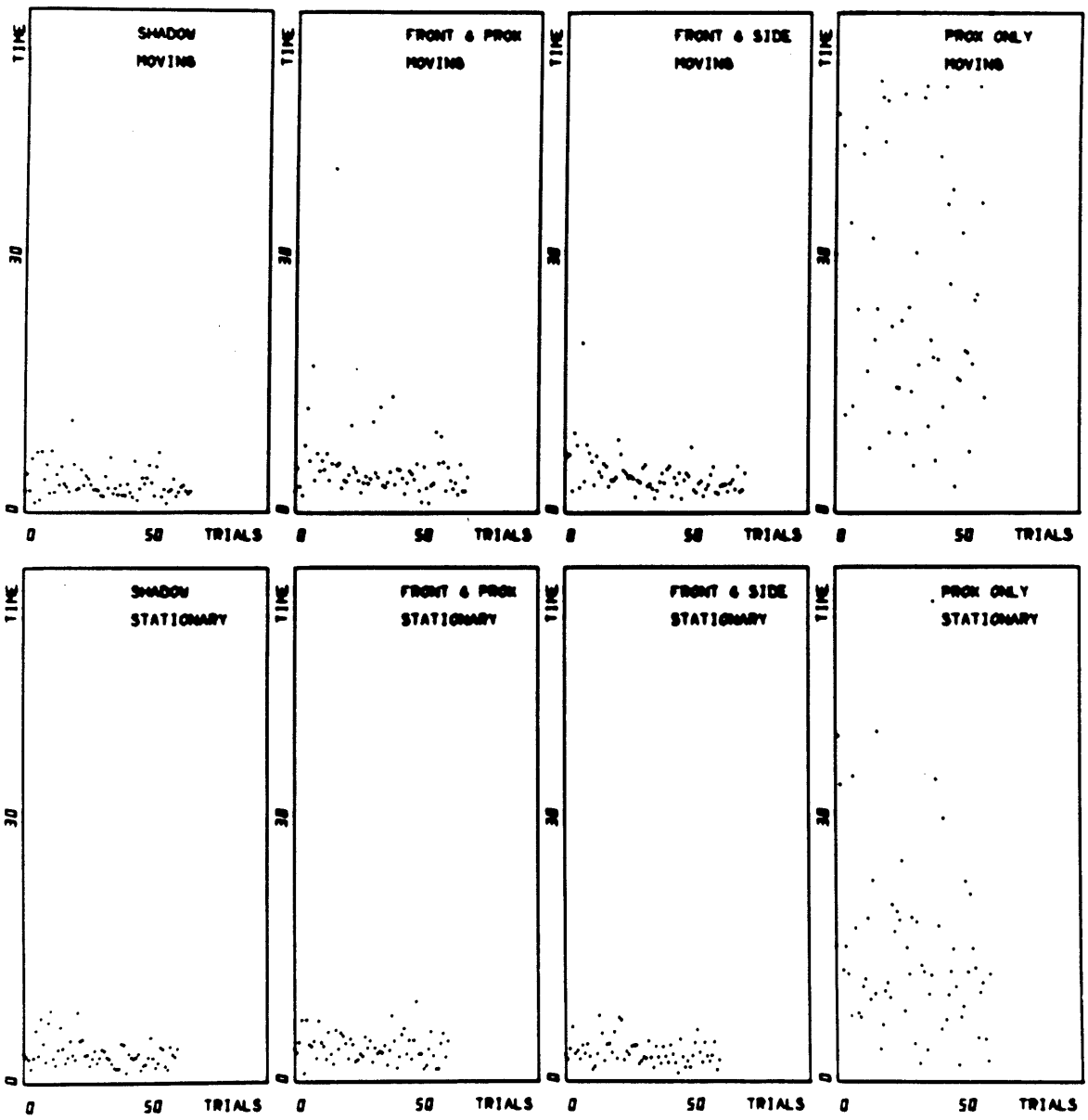

Figure 24 Learning Curves - Subject 1

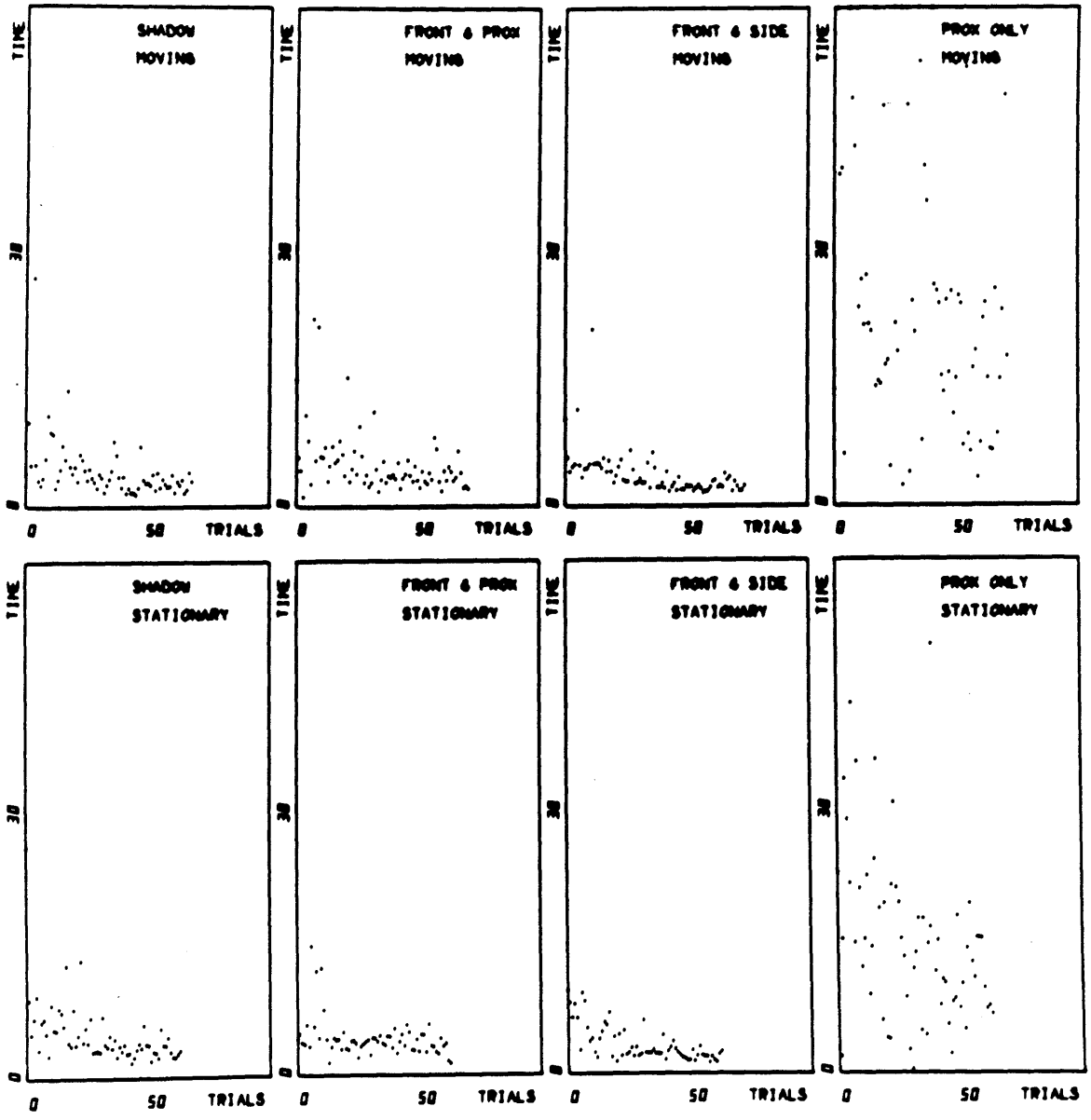

Figure 25 Learning Curves - Subject 2

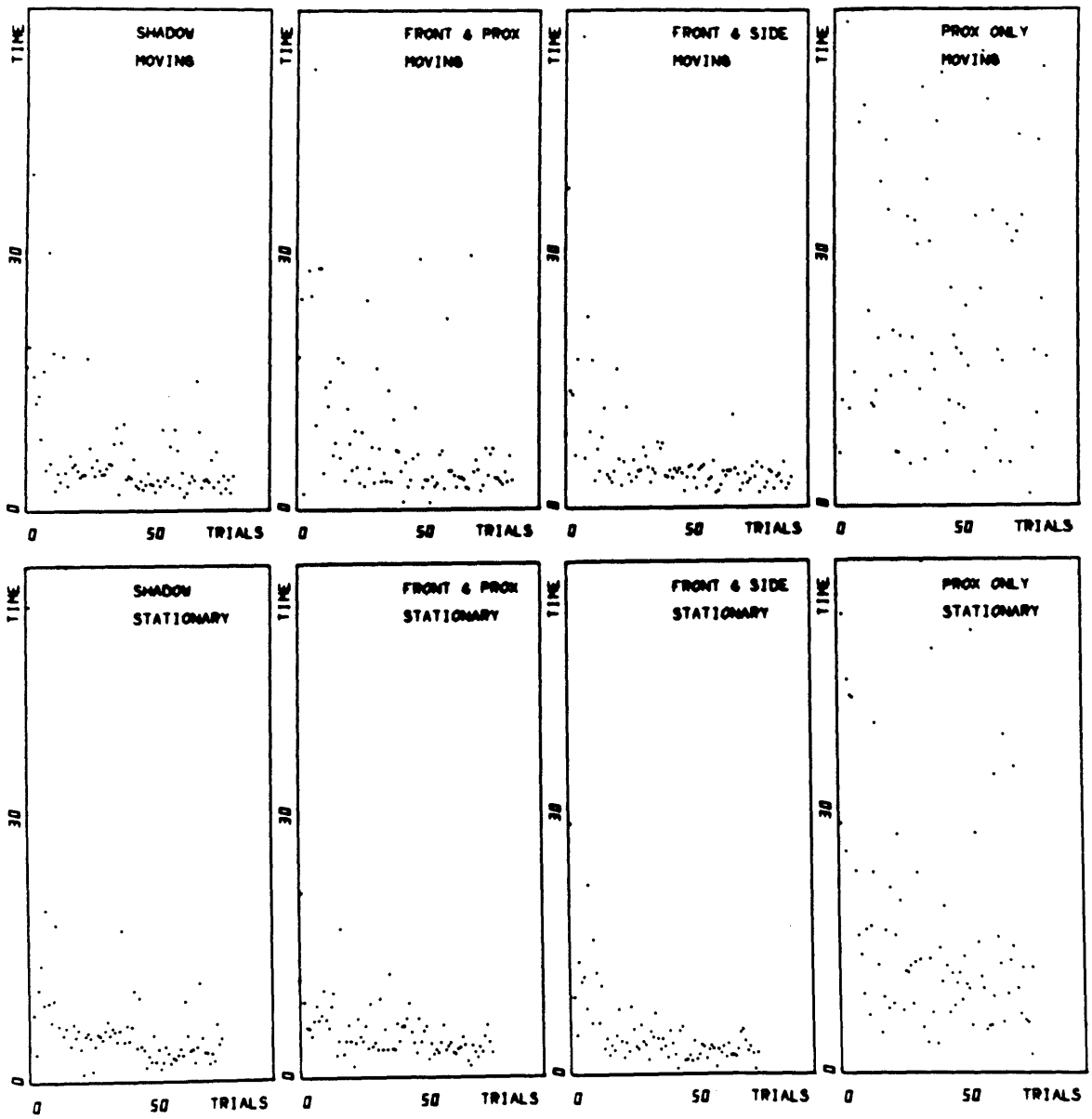

Figure 26 Learning Curves - Subject 3

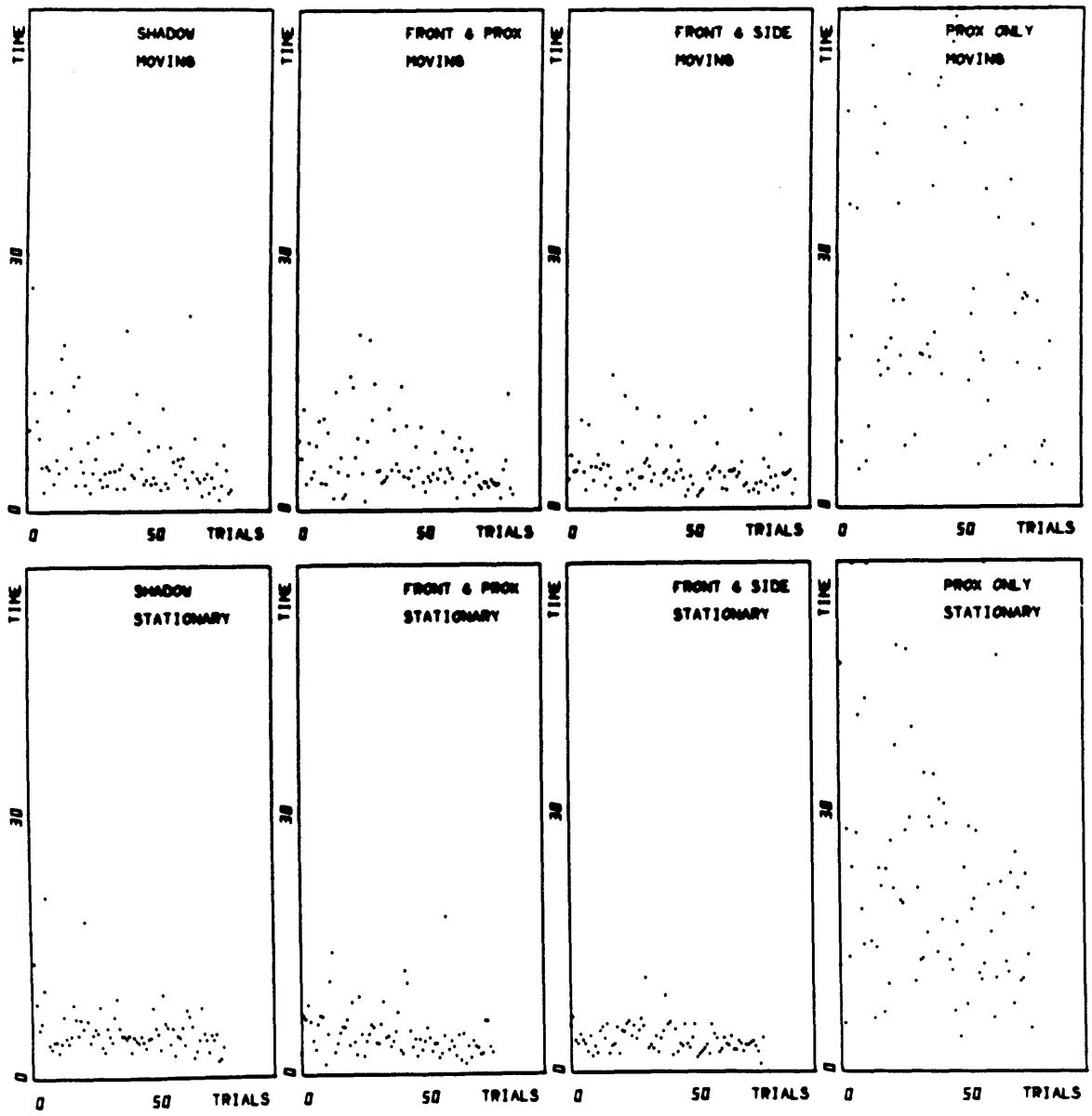

Figure 27 Learning Curves - Subject 4

TABLE 3 AVERAGE TIME TO LOCATE OBJECT (seconds)
SUBJECT #3

DISPLAY \ TASK	stationary		moving		both tasks	
	$\bar{X}$	S	$\bar{X}$	S	$\bar{X}$	S
shadow	3.74	2.05	4.74	2.93	4.24	2.58
front & prox	3.73	1.26	5.56	5.73	4.65	4.43
front & side	2.83	1.11	3.77	1.67	3.31	1.49
prox only	14.07	11.21	35.36	28.51	24.72	24.14
all displays	6.09	7.38	12.36	22.37	10.15	17.26

TABLE 4 AVERAGE TIME TO LOCATE OBJECT (seconds)
SUBJECT #4

DISPLAY \ TASK	stationary		moving		both tasks	
	$\bar{X}$	S	$\bar{X}$	S	$\bar{X}$	S
shadow	4.93	1.83	5.12	4.00	5.03	3.11
front & prox	3.91	2.90	4.43	2.51	4.17	2.74
front & side	3.26	.99	4.28	2.39	3.77	1.90
prox only	20.90	11.21	34.44	36.14	27.67	27.60
all displays	8.24	9.40	12.07	22.37	10.15	17.26

TABLE 5 AVERAGE TIME TO LOCATE OBJECT (seconds)
ALL SUBJECTS

DISPLAY \ TASK	stationary		moving		both tasks	
	$\bar{X}$	S	$\bar{X}$	S	$\bar{X}$	S
shadow	3.73	1.54	4.00	2.67	3.87	2.18
front & prox	4.06	1.89	4.37	3.29	4.22	2.69
front & side	2.88	.99	3.50	1.65	3.19	1.40
prox only	15.26	14.33	31.47	27.69	23.36	23.49
all displays	6.62	8.88	10.8	18.41	8.94	14.62

From looking at the results of the experiment it was clear that the proximity indicator alone was a much poorer depth indicator than the other three displays. In order that the poor results from the proximity indicator not overwelm the other indicators, it was left out from further analysis. A three-way analysis of variance was performed on the remaining three displays using IBM's Scientific Subroutine

Package (ANOVA) which computes the mean square for each variation and interaction. The signifacance of the results was determined by a F statistic defined as the mean square of the variation in question divided by the mean square the residuals. The residuals were taken to be all interactions involving the number of trails. The following is a condensation of the results.

TABLE 6 THREE-WAY ANALYSIS OF VARIANCE

S=SUBJECTS
E=EXPERIMENT TYPE (MOVING OR STATIONARY)
D=DISPLAY TYPE (SHADOW, FRONT & SIDE, FRONT & PROX)
R=REPETITIONS

SOURCE OF VARIATION	F(f1,f2)	LEVEL OF CONFIDENCE (nul)
S	13.26(3,744)	.0005
E	9.84(1,744)	.001
D	11.70(2,744)	.0005
SE	3.58(3,744)	.125
SD	2.98(6,744)	.01
ED	.50(2,744)	.95
SED	.36(6,744)	.95

These results show clear differences between subjects, display types and the two experiments. There are significant differences among individual subjects with respect to their performance both on the different display types and on experiments. The experiment type had no signifacant effect on differences between the display types.

APPENDIX II

SIMULATION TRANSFORMATION MATRICES

The arm was divided into three sections: the hand, the forearm and the upper arm. The coordinates of the center of rotation of each section were set at the origin. This eliminated the need to translate the center of rotation to the origin, perform the rotation, then translate the center of rotation back to the proper position. Instead the section in question needed only to be rotated, then translated to the proper location. The points of rotation and the designation of the angles are shown in figure 12. At each joint, the order of rotation was torsion first, followed by flexure. Rotations were performed first at the extremities. First the hand was rotated, then it was translated so that its center of rotation was located at the end of the forearm. Next the hand and the forearm were rotated about the elbow, then the hand and the forearm were translated so that the elbow was located at the end of the upper arm. Lastly the whole arm was rotated about the shoulder. For simplicity, cos(An) and sin(An) were abbreviated Cn and Sn respectively. Transformation matrices were found for each section of the arm. For the convenience of the reader, the notation is the same as that used in the simulation software listed in Appendices V and VI.

The rotation matrix of the upper arm around the

shoulder is given by:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C1 & -S1 & 0 \\ 0 & S1 & C1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} C2 & -S2 & 0 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Which reduces to:

$$\begin{bmatrix} C2 & -S2 & 0 & 0 \\ C1S2 & C1C2 & -S1 & 0 \\ S1S2 & S1C2 & C1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This was abbreviated as:

$$\begin{bmatrix} XX & XY & 0 & 0 \\ YX & YY & YZ & 0 \\ ZX & ZY & ZZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for rotation about the elbow and translation to the end of the upper arm is found from:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & ZFT \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C3 & -S3 & 0 \\ 0 & S3 & C3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C4 & -S4 & 0 & 0 \\ S4 & C4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which reduces to:

$$\begin{bmatrix} C4 & -S4 & 0 & 0 \\ C3S4 & C3C4 & -S3 & 0 \\ S3S4 & S3C4 & C3 & ZFT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This transformatiom matrix multiplied with the matrix for

rotation about the shoulder calculated above gives the following transformation matrix for rotation of the forearm:

$$\begin{bmatrix} XXC4+XYC3S4 & -XXS4+XYC3C4 & -XYS3 & 0 \\ YXC4+YYC3S4+YZS3S4 & -YXS4+YYC3C4+YZS3C4 & -YYS3+YZC3 & YZZFT \\ ZXC4+ZYC3S4+ZZS3S4 & -ZXS4+ZYC3C4+ZZS3C4 & -ZYS3+ZZC3 & ZZZFT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For simplicity this was abbreviated as:

$$\begin{bmatrix} XX1 & XY1 & XZ1 & 0 \\ YX1 & YY1 & YZ1 & YT1 \\ ZX1 & ZY1 & ZZ1 & ZT1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for rotation of the hand about the wrist and translation of the wrist to the end of the hand is determined in the same manner as the transformation matrix for the forearm. It is given by:

$$\begin{bmatrix} C6 & -S6 & 0 & 0 \\ C5S6 & C5C6 & -S5 & 0 \\ S5S6 & S5C6 & C5 & ZST \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is multiplied by the transformation matrix for rotation about the elbow and shoulder to give:

$$\begin{bmatrix} XX1C6+XY1C5S6+XZ1S5S6 & -XX1S6+XY1C5C6+XZ1S5C6 & -XY1S5+XZ1C5 & XZ1ZST \\ YX1C6+YY1C5S6+YZ1S5S6 & -YX1S6+YY1C5C6+YZ1S5C6 & -YY1S5+YZ1C5 & YZ1ZST+YT1 \\ ZX1C6+ZY1C5S6+ZZ1S5S6 & -ZX1S6+ZY1C5C6+ZZ1S5C6 & -ZY1S5+ZZ1C5 & ZZ1ZST+ZT1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For simplicity, this was abbreviated as:

$$\begin{bmatrix} XX2 & XY2 & XZ2 & XT2 \\ YX2 & YY2 & YZ2 & YT2 \\ ZX2 & ZY2 & ZZ2 & ZT2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The translation matrix for the closure of the tongs for a given closure coefficient D is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1.9D \\ 0 & 0 & 1 & -.8D \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The motion of the submarine is described by roll about the z-axis, followed by pitching about the x-axis and finally yaw about the y-axis as expressed by:

$$\begin{bmatrix} CV2 & 0 & SV2 & 0 \\ 0 & 1 & 0 & 0 \\ -SV2 & 0 & CV2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & CV1 & -SV1 & 0 \\ 0 & SV1 & CV1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} CV3 & -SV3 & 0 & 0 \\ SV3 & CV3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The vehicle is then moved into position by the translation:

$$\begin{bmatrix} 1 & 0 & 0 & V11 \\ 0 & 1 & 0 & V12+YSUBT \\ 0 & 0 & 1 & V10+ZSUBT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The resulting transformation matrix is:

$$\begin{bmatrix} CV1CV2+SV1SV2SV3 & -SV1CV2+CV1SV2SV3 & SV2CV3 & V11 \\ SV1CV3 & CV1CV3 & -SV3 & V12+YSUBT \\ -CV1SV2+SV1CV2SV3 & SV1SV2+CV1CV2SV3 & CV2CV3 & V10+ZSUBT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The submarine transformation matrix was abbreviated by:

$$\begin{bmatrix} SXX & SXY & SXZ & SXT \\ SYX & SYY & SYZ & SYT \\ SZX & SZY & SZZ & SZT \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With the above transformations, the arm can be completely described in three-space with seven degrees-of-freedom and the vehicle motion can be described in six degrees-of-freedom.

The azimuth and zenith of the viewpoint were selectable by the operator. The order of operations was an azimuth rotation about the y-axis, followed by a zenith transformation about the x-axis, and finally a positioning translation incorporating a scaling factor S. These transformations are given by:

$$\begin{bmatrix} S & 0 & 0 & STX \\ 0 & S & 0 & STY \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & WC2 & -WS2 & 0 \\ 0 & WS2 & WC2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} WC1 & 0 & WS1 & 0 \\ 0 & 1 & 0 & 0 \\ -WS1 & 0 & WC1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the graphic terminal only displays X and Y, only the first two rows needed to be computed yielding:

$$\begin{bmatrix} SWC1 & 0 & SWS1 & STX \\ SWS1WS2 & SWC2 & -SWC1WS2 & STY \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These transformations were abbreviated by:

$$\begin{bmatrix} WXX & WXY & WXZ & WXT \\ WYX & WYY & WYZ & WYT \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

APPENDIX III

FORCE-FEEDBACK JACOBIAN CALCULATIONS

In order to implement simulated force-feedback, the jacobian matrix for the tip of the tongs needed to be calculated. The position of the tip of the tongs is given by:

$$\begin{bmatrix} XX2 & XY2 & XZ2 & XT2 \\ YX2 & YY2 & YZ2 & YT2 \\ ZX2 & ZY2 & ZZ2 & ZT2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ ZHT \\ 1 \end{bmatrix}$$

This reduces to the position vector for the tip of the tongs given by:

$$\begin{bmatrix} XZ2ZHT+XT2 \\ YZ2ZHT+YT2 \\ ZZ2ZHT+ZT2 \\ 1 \end{bmatrix}$$

The derivatives of this position vector are the jacobian matrix. In the simulation software, this matrix was represented by:

$$\begin{bmatrix} DXD1 & DXD2 & DXD3 \\ DYD1 & DXD2 & DYD3 \\ DZD1 & DZD2 & DZD3 \end{bmatrix}$$

The terms of this matrix are given by:

```
DXD1=0
DYD1=(-S1S2S4+S1C2C3C4+C1S3C4)S5HTR
     +(S1C2S3-C1C3)(C5HTR+ZFT)-C1ZST
DZD1=(C1S2S4-C1C2C3C4+S1S3C4)S5HTR
     +(-C1C2S3-S1C3)(C5HTR+ZFT)-S1ZST
DXD2=(-S2S4+C2C3C4)S5HTR+C2S3(C5HTR+ZFT)
DYD2=(C1C2S4+C1S2C3C4)S5HTR+C1S2S3(C5HTR+ZFT)
DZD2=(S1C2S4+S1S2C3C4)S5HTR+S1S2S3(C5HTR+ZFT)
```

```
DXD3=-S2S3C4S5HTR+S2C3(C5HTR+ZFT)
DYD3=(C1C2S3C4+S1C3C4)S5HTR+(-C1C2C3+S1S3)(C5HTR+ZFT)
DZD3=(S1C2S3C4-C1C3C4)S5HTR+(-S1C2C3-C1S3)(C5HTR+ZFT)
```

The jacobian matrix for the manipulator attached to a vehicle is found by premultiplying the jacobian matrix determined above by the rotation matrix for the vehicle.

$$\begin{bmatrix} DXD1' & DXD2' & DXD3' \\ DYD1' & DYD2' & DYD3' \\ DZD1' & DZD2' & DZD3' \end{bmatrix} = \begin{bmatrix} SXX & SXY & SXZ \\ SYX & SYY & SYZ \\ SZX & SZY & SZZ \end{bmatrix} \begin{bmatrix} DXD1 & DXD2 & DXD3 \\ DYD1 & DYD2 & DYD3 \\ DZD1 & DZD2 & DZD3 \end{bmatrix}$$

APPENDIX IV

AUXILLARY SOFTWARE

This program calls several subroutines involved in performing graghics on the Megatek graphics terminal.

MGINIT initializes the graphics terminal.

NPOINT(loc) reads the address of the next entry into the display.

MOVI3(ix,iy,iz) moves the beam to the position ix, iy, iz without drawing a line.

DRWI3(ix,iy,iz) draws a line from the previous position of the beam to ix, iy, iz.

RMOVI3(ix,iy,iz) moves the beam an increment ix, iy, iz from its last position.

RDRWI3(ix,iy,iz) draws a line an increment ix, iy, iz from the last beam position.

MGSEND sends the picture to the Megatek display.

MODIFY(loc) replaces the command at address loc of the display list by the next Megatek command executed.

LDTRN3(xx,xy,xz,xt,yx,yy,yz,yt) loads the first two rows of a transformation matrix into the hardware rotate.

SUBDEF(i) defines beginning of subpicture i.

SUBEND(i) defines the end of subpicture i.

SUBCAL(i) calls subpicture i.

The following programs operate the A/D converter:

ANINIT initializes the A/D converter.

AINSQ(ichan1,ichan2,iarray) reads data from analog input channels ichan1 to ichan2 of the A/D converter into the array iarray.

AOUTSQ(ichan1,ichan2,iarray) writes from array iarray to analog output channels ichan1 to ichan2.

DIN(ichan,ivar) reads digital input channel ichan into ivar.

DOUT(ichan,ivar) writes ivar to digital output channel ichan.

PATH, NPATH are arbitrary contours which are used by H. Kazerooni's dynamic programs for bottom following experiments.

GETADR, READEF AND QIO allow the keyboard to be read with out interrupting the program.

APPENDIX V

MANIPULATOR SIMULATION SOFTWARE

```
C INITIALIZE PROGRAM
        DIMENSION IA(16),A(7),IPARAM(6),IOSB(2),IBUF(12)
        BYTE ACON(34,2)
        COMMON /DMABUF/IDUM(3610),ADAT(22,3),PNT(22,2),
     1  TCH(3,6),SCL(7,2),II(7),I1(7),I2(7),I3(7),IS(7),F1,
     1  IT1,IT2,ICTL
        DATA OXX,OXY,OXZ/1.,0.,0./
        DATA OYX,OYY,OYZ/0.,1.,0./
        DATA OZX,OZY,OZZ/0.,0.,1./
        ICTL=0
100     WRITE(5,1)
1       FORMAT(' ENTER FUNCTION'/' 1=SPHERE'/' 2=OBJECT'/
     1  ' 4=SURFACE')
        READ(5,*)IFUNC
        ISPH=0
        IOBJ=0
        ISURF=0
        IF((IFUNC.AND."1).NE.0)ISPH=1
        IF((IFUNC.AND."2).NE.0)IOBJ=1
        IF((IFUNC.AND."4).NE.0)ISURF=1
C INITIALIZE THE MEGATEK
        CALL MGINIT
        CALL SETINT(13)
C INITALIZE THE A/D
        CALL ANINIT
        CALL AINSQ(16,29,IA)
        CALL AOUTSQ(4,17,IA)
C READ MANIPULATOR CONTROL PARAMETERS
        WRITE(5,2)
2       FORMAT(' FEEDFORWARD ACTIVE AND PASSIVE CYCLE TIMES
     1  (.7,30,150)')
        READ(5,*)F1,IT1,IT2
        ICTL=1
        CALL DOUT(24,"77)
C INITIALIZE THE KEYBOARD MONITER ROUTINE
        CALL GETADR(IPARAM(1),ICMD)
        IPARAM(2)=1
C SET FUNCTION FLAGS
        ISHAD=-1          !SHADOW
        IWALL=-1          !WALLS
        ITCHPT=-1         !TOUCHPOINTS
        ICOOR=-1          !COORDINATES
        IPROX=-1          !PROXIMITY INDICATOR
C SET POSITION OF ARM PARTS
        ZHC=-4.4          !CENTER OF HAND
```

```
        ZHT=-5.6          !END OF HAND
        ZFT=-40.3         !END OF FOREARM
        ZST=-18.4         !END OF SHOULDER
        SF=40.            !DISPLAY SCALE
        NHL=34            !# LINES IN HAND
        NSL=16            !# LINES IN SHOULDER
        NFL=20            !# LINES IN FORARM
C SET POSITION OF OBJECTS
        XO=0
        YO=-40
        ZO=-24
        OXT=-10
        OYT=-35
        OZT=0
        IFLG=0
        IFLGO=0
        IFLGS=0
C SET INITIAL VELOCITY TO ZERO
        VX=0.
        VY=0.
        VZ=0.
        OVX=0
        OVY=0
        OVZ=0
        TIM=SECNDS(0.)
C SET VIEWPOINT AND ZOOM
        TX=0.
        TY=0.
        S=1
        WA1=0
        WA2=0
        DWA1=0
        DWA2=0
        DTX=0
        DTY=0
        DS=0
C READ SCALING FACTORS FOR A/D OUTPUT OF ANGLES
        OPEN(UNIT=4,NAME='DL1:[200,125]SCALE.DAT',TYPE='OLD')
        READ(4,*)((SCL(I,J),J=1,2),I=1,7)
        CLOSE(UNIT=4,DISPOSE='SAVE')
C READ ARM POINT DATA
        OPEN(UNIT=4,NAME='DL1:[200,125]ARMDAT.DAT',TYPE='OLD')
        READ(4,*)((ADAT(I,J),J=1,3),I=1,22)
        CLOSE(UNIT=4,DISPOSE='SAVE')
C READ ARM CONNECTIVITY DATA
        OPEN(UNIT=4,NAME='DL1:[200,125]ARMCON.DAT',TYPE='OLD')
        READ(4,*)((ACON(I,J),J=1,2),I=1,NHL)
        CLOSE(UNIT=4,DISPOSE='SAVE')
C READ OBJECT TOUCHING CONDITIONS
```

```
        OPEN(UNIT=4,NAME='DL1:[200,125]OBJTCH.DAT',TYPE='OLD')
        DO 101 I=1,5
        NOP=I-1
        READ(4,*,END=102)(TCH(I,J),J=1,6)
101     CONTINUE
102     CLOSE(UNIT=4,DISPOSE='SAVE')
C READ PARAMETERS OF SURFACE
        IF(ISURF.NE.1)GOTO 103
        WRITE(5,3)
3       FORMAT(' ENTER PARAMETERS P1,P2 OF SURFACE (200,300)')
        READ(5,*)P1,P2
C READ DESIRED DECCELLERATION, GRAVITY, ELASTICITY
103     IF((IOBJ.NE.1).AND.(ISPH.NE.1))GOTO 104
        WRITE(5,4)
4       FORMAT(' DECCELERATION, GRAVITY, STIFFNESS,
     1  ELASTICITY (0<<1)')
        READ(5,*)DRAG,G,AB,ELAST
C PRINT DISPLAY CONTROL INSTRUCTIONS
104     WRITE(5,5)
5       FORMAT(' TO ROTATE TYPE:')
        WRITE(5,6)
6       FORMAT(' 8=UP  2=DOWN  6=RIGHT  4=LEFT  5=STOP
     1  1=FRONT  7=SIDE')
        WRITE(5,7)
7       FORMAT(' TO TRANSLATE TYPE:')
        WRITE(5,8)"047  !QUOTE
8       FORMAT(' "="=UP  ',A,'=DOWN  [=RIGHT
     1  ®=LEFT  0=CENTER')
        WRITE(5,9)
9       FORMAT(' TO ZOOM TYPE:')
        WRITE(5,10)
10      FORMAT(' 9=ZOOM UP  3=ZOOM DOWN')
        WRITE(5,11)
11      FORMAT(' TO DRAW LINES TYPE:')
        WRITE(5,12)
12      FORMAT(' M=MOVE  D=DRAW  E=ERASE LAST LINE')
        WRITE(5,13)
13      FORMAT(' TYPE "S" FOR SHADOW, "W" FOR WALLS'/
     1  ' "T" FOR TOUCHPOINTS, "P" FOR PROXIMITY')
        WRITE(5,14)
14      FORMAT(' TYPE "Z" TO EXIT, "R" TO REPEAT')
C LOAD SUBPICTURES
C LOAD WALLS IN SUB 1
        CALL SUBDEF(1)
        I=SF*.75
        J=SF*12.75
        K=SF*26.75
        L=SF*36.75
        M=SF*48.75
```

```
          CALL MOVI3(K,I,J)
          CALL DRWI3(K,-M,J)
          CALL DRWI3(-K,-M,J)
          CALL DRWI3(-K,I,J)
          CALL DRWI3(K,I,J)
          CALL DRWI3(K,I,-L)
          CALL DRWI3(K,-M,-L)
          CALL DRWI3(-K,-M,-L)
          CALL DRWI3(-K,I,-L)
          CALL DRWI3(K,I,-L)
          CALL MOVI3(K,-M,-L)
          CALL DRWI3(K,-M,J)
          CALL MOVI3(-K,-M,J)
          CALL DRWI3(-K,-M,-L)
          CALL MOVI3(-K,I,-L)
          CALL DRWI3(-K,I,J)
          CALL SUBEND(1)
          OPEN(UNIT=1,NAME='DL1:[200,125]ARMCON.DAT',TYPE='OLD')
          OPEN(UNIT=2,NAME='DL1:[200,125]ARMDAT.DAF',TYPE='OLD'
      1   ,ACCESS='DIRECT',RECORDSIZE=3)
C SPOOL THROUGH FILE TO FOREARM DATA
          DO 105 I=1,NHL
          READ(1,*)M,MM
105       CONTINUE
C LOAD FOREARM IN SUB 2
          CALL SUBDEF(2)
          CALL DRAW(NFL,SF)
          CALL SUBEND(2)
C LOAD SHOULDER IN SUB 3
          CALL SUBDEF(3)
          CALL DRAW(NSL,SF)
          CALL SUBEND(3)
          CLOSE(UNIT=1,DISPOSE='SAVE')
          CLOSE(UNIT=2,DISPOSE='SAVE')
C LOAD SPHERE IN SUB 4
          CALL SUBDEF(4)
          I=SF*.525
          J=SF*.225
          CALL RMOVI3(0,IFIX(.75*SF),0)
          CALL RDRWI3(I,-J,0)
          CALL RDRWI3(J,-I,0)
          CALL RDRWI3(-J,-I,0)
          CALL RDRWI3(-I,-J,0)
          CALL RDRWI3(-I,J,0)
          CALL RDRWI3(-J,I,0)
          CALL RDRWI3(J,I,0)
          CALL RDRWI3(I,J,0)
          CALL SUBEND(4)
C LOAD OBJECT INTO SUB 5
```

```
      OPEN(UNIT=1,NAME='DL1:[200,125]OBJCON.DAT',TYPE='OLD')
      OPEN(UNIT=2,NAME='DL1:[200,125]OBJDAT.DAF',TYPE='OLD'
     1  ,ACCESS='DIRECT',RECORDSIZE=3)
      CALL SUBDEF(5)
      CALL DRAW(100,SF)
      CALL SUBEND(5)
      CLOSE(UNIT=1,DISPOSE='SAVE')
      CLOSE(UNIT=2,DISPOSE='SAVE')
C LOAD PERMANANT DISPLAY INTO SUB 6
      CALL SUBDEF(6)
C LOAD SURFACE
      IF(ISURF.NE.1)GOTO 110
      CALL SETINT(10)
      DO 107 I=-2000,2000,200
      Z=I
      X=-2000.
      IY=SF*YSURF(X/SF,Z/SF,P1,P2)
      CALL MOVI3(-2000,IY,I)
      DO 106 J=-2000,2000,200
      X=J
      IY=SF*YSURF(X/SF,Z/SF,P1,P2)
      CALL DRWI3(J,IY,I)
106   CONTINUE
      CALL MGSEND
107   CONTINUE
      DO 109 I=-2000,2000,200
      X=I
      Z=-2000.
      IY=SF*YSURF(X/SF,Z/SF,P1,P2)
      CALL MOVI3(I,IY,-2000)
      DO 108 J=-2000,2000,200
      Z=J
      IY=SF*YSURF(X/SF,Z/SF,P1,P2)
      CALL DRWI3(I,IY,J)
108   CONTINUE
      CALL MGSEND
109   CONTINUE
      CALL SETINT(13)
C SAVE MEMORY SPACE FOR INPUTTING LINES
110   CALL NPOINT(LINES)
      DO 111 I=1,40
      CALL MOVI3(0,0,0)
111   CONTINUE
      CALL NPOINT(LSHADS)
      DO 112 I=1,40
      CALL MOVI3(0,0,0)
112   CONTINUE
      CALL SUBEND(6)
C LOAD INITIAL ARM POSITION
```

```
        DO 113 I=1,22
        PNT(I,1)=ADAT(I,2)
        PNT(I,2)=ADAT(I,3)
113     CONTINUE
        CALL NPOINT(LOC)
C READ ARM POSITION FROM A/D CONVERTER AND CONVERT TO VOLTAGE
114     CALL AINSQ(23,29,IS)
        DO 115 I=1,7
        I3(I)=IS(I)
        A(I)=FLOAT(IS(I))/3276.2
115     CONTINUE
C INCREMENT VIEWPOINT
        WA1=WA1+DWA1
        WA2=WA2+DWA2
        S=S+DS
        TX=TX+DTX
        TY=TY+DTY
C SCALE A/D OUTPUT AND CALCULATE SINES & COSINES
        AH=SCL(1,1)*(A(5)+SCL(1,2))
        S1=SIN(AH)
        C1=COS(AH)
        AH=SCL(2,1)*(A(7)+SCL(2,2))
        S2=SIN(AH)
        C2=COS(AH)
        AH=SCL(3,1)*(A(6)+SCL(3,2))-SCL(1,1)*(A(5)+SCL(1,2))
        S3=SIN(AH)
        C3=COS(AH)
        AH=SCL(4,1)*(A(2)+SCL(4,2))
        S4=SIN(AH)
        C4=COS(AH)
        AH=SCL(5,1)*((A(3)-A(4))/2.+SCL(5,2))
        S5=SIN(AH)
        C5=COS(AH)
        AH=SCL(6,1)*((A(3)+A(4))/2.+SCL(6,2))
        S6=SIN(AH)
        C6=COS(AH)
        D=SCL(7,1)*(A(1)+SCL(7,2))
        WS1=SIN(WA1)
        WC1=COS(WA1)
        WS2=SIN(WA2)
        WC2=COS(WA2)
        HCR=ZHC-.8*D
        HTR=ZHT-.8*D
C CALCALATE TRANSFORMATIONS
C CALCULATE VIEWPOINT TRANSFORMATIONS
        WXX=WC1*S
        WXY=0
        WXZ=WS1*S
        WXT=TX*S
```

```
          WYX=WS1*WS2*S
          WYY=WC2*S
          WYZ=-WC1*WS2*S
          WYT=TY*S
C CALCULATE dX/dA1
          YY=S1*S2*S4-S1*C2*C3*C4-C1*S3*C4
          YZ=S1*C2*S3-C1*C3
          YT=-C1*ZST
          ZY=-C1*S2*S4+C1*C2*C3*C4-S1*S3*C4
          ZZ=-C1*C2*S3-S1*C3
          ZT=-S1*ZST
          DYD1=(-YY*S5+YZ*C5)*HTR+YZ*ZFT+YT
          DZD1=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT+ZT
C CALCULATE dX/dA2
          XY=S2*S4-C2*C3*C4
          XZ=C2*S3
          YY=-C1*C2*S4-C1*S2*C3*C4
          YZ=C1*S2*S3
          ZY=-S1*C2*S4-S1*S2*C3*C4
          ZZ=S1*S2*S3
          DXD2=(-XY*S5+XZ*C5)*HTR+XZ*ZFT
          DYD2=(-YY*S5+YZ*C5)*HTR+YZ*ZFT
          DZD2=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT
C CALCULATE dX/dA3
          XY=S2*S3*C4
          XZ=S2*C3
          YY=-C1*C2*S3*C4-S1*C3*C4
          YZ=-C1*C2*C3+S1*S3
          ZY=-S1*C2*S3*C4+C1*C3*C4
          ZZ=-S1*C2*C3-C1*S3
          DXD3=(-XY*S5+XZ*C5)*HTR+XZ*ZFT
          DYD3=(-YY*S5+YZ*C5)*HTR+YZ*ZFT
          DZD3=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT
C CALCULATE INVERSE
          DET=DXD2*DYD3*DZD1+DXD3*DYD1*DZD2
     1    -DZD1*DYD2*DXD3-DYD1*DXD2*DZD3
          R11=(DYD2*DZD3-DYD3*DZD2)/DET
          R12=(DZD2*DXD3-DXD2*DZD3)/DET
          R13=(DXD2*DYD3-DYD2*DXD3)/DET
          R21=(DYD3*DZD1-DYD1*DZD3)/DET
          R22=-DXD3*DZD1/DET
          R23=DYD1*DXD3/DET
          R31=(DYD1*DZD2-DYD2*DZD1)/DET
          R32=(DXD2*DZD1-DZD2*DXD1)/DET
          R33=-DXD2*DYD1/DET
C SHOULDER TRANSFORMATIONS
          XX=C2
          XY=-S2
          YX=C1*S2
```

```
          YY=C1*C2
          YZ=-S1
          ZX=S1*S2
          ZY=S1*C2
          ZZ=C1
C FOREARM TRANSFORMATIONS
          XX1=XX*C4+XY*C3*S4
          XY1=-XX*S4+XY*C3*C4
          XZ1=-XY*S3
          YX1=YX*C4+YY*C3*S4+YZ*S3*S4
          YY1=-YX*S4+YY*C3*C4+YZ*S3*C4
          YZ1=-YY*S3+YZ*C3
          YT1=YZ*ZST
          ZX1=ZX*C4+ZY*C3*S4+ZZ*S3*S4
          ZY1=-ZX*S4+ZY*C3*C4+ZZ*S3*C4
          ZZ1=-ZY*S3+ZZ*C3
          ZT1=ZZ*ZST
C HAND TRANSFORMATIONS
          XX2=XX1*C6+XY1*C5*S6+XZ1*S5*S6
          XY2=-XX1*S6+XY1*C5*C6+XZ1*S5*C6
          XZ2=-XY1*S5+XZ1*C5
          XT2=XZ1*ZFT
          YX2=YX1*C6+YY1*C5*S6+YZ1*S5*S6
          YY2=-YX1*S6+YY1*C5*C6+YZ1*S5*C6
          YZ2=-YY1*S5+YZ1*C5
          YT2=YZ1*ZFT+YT1
          ZX2=ZX1*C6+ZY1*C5*S6+ZZ1*S5*S6
          ZY2=-ZX1*S6+ZY1*C5*C6+ZZ1*S5*C6
          ZZ2=-ZY1*S5+ZZ1*C5
          ZT2=ZZ1*ZFT+ZT1
C FIND ELAPSED TIME
          OLDTIM=TIM
          TIM=SECNDS(0.)
          DELTIM=TIM-OLDTIM
C DETERMINE POINT MIDWAY BETWEEN JAWS OF HAND
          XC=XZ2*HCR+XT2
          YC=YZ2*HCR+YT2
          ZC=ZZ2*HCR+ZT2
          XT=XZ2*HTR+XT2
          YT=YZ2*HTR+YT2
          ZT=ZZ2*HTR+ZT2
C DETERMINE IF HAND IS HOLDING SPHERE
          IF(ISPH.NE.1)GOTO 118
C IF GRIPPED LAST CYCLE AND GRIPS STILL CLOSED
          IF((IFS.EQ.1).AND.(D.GE..6))GOTO 116
          IF((ABS(XC-XO).GT.1.).OR.(ABS(YC-YO).GT.1.).OR.
     1   (ABS(ZC-ZO).GT.1.).OR.(D.LT..6))GOTO 117
C IF HAND IS HOLDING SPHERE
C SET VELOCITY OF SPHERE TO VELOCTY OF HAND
```

```
        IFS=1
116     VX=(XC-XO)/DELTIM
        VY=(YC-YO)/DELTIM
        VZ=(ZC-ZO)/DELTIM
C SET CENTER OF SPHERE TO POSITION OF HAND
        XO=XC
        YO=YC
        ZO=ZC
C HOLD HAND OPEN AROUND SPHERE
        D=.6
C SET FORCE FEEDBACK FLAG
        IFORCE=1
        GOTO 118
C IF HAND IS NOT HOLDING SPHERE
117     IFORCE=0
        IFS=0
C KEEP SPHERE WITHIN BOUNDARIES
        IF(ABS(XO).GT.26.)VX=-VX*ELAST
        IF(ABS(YO+24.).GT.24.)VY=-VY*ELAST
        IF(ABS(ZO+12.).GT.24.)VZ=-VZ*ELAST
C ACCELERATE/DECCELERATE SPHERE
        XO=XO+VX*DELTIM
        YO=YO+VY*DELTIM
        ZO=ZO+VZ*DELTIM
        IF(YO.GT.-48)VY=VY-G*DELTIM
        VX=VX*(1-DRAG)
        VY=VY*(1-DRAG)
        VZ=VZ*(1-DRAG)
        PROX=SQRT((XO-XC)**2+(YO-YC)**2+(ZO-ZC)**2)
C DETERMINE IF HAND IS HOLDING OBJECT
118     IF(IOBJ.NE.1)GOTO 123
        IF((IFO.EQ.1).AND.(DD.LE.D))GOTO 121
C FIND HAND REFERENCE TO OBJECT REFERENCE TRANSFORMATION
        DET=OXX*OYY*OZZ+OXY*OYZ*OZX+OXZ*OYX*OZY
     1  -OZX*OYY*OXZ-OZY*OYZ*OXX-OZZ*OYX*OXY
        RXX=(OYY*OZZ-OZY*OYZ)/DET
        RXY=(OZY*OXZ-OXY*OZZ)/DET
        RXZ=(OXY*OYZ-OYY*OXZ)/DET
        RXT=-RXX*OXT-RXY*OYT-RXZ*OZT
        RYX=(OYZ*OZX-OYX*OZZ)/DET
        RYY=(OXX*OZZ-OXZ*OZX)/DET
        RYZ=(OYX*OXZ-OYZ*OXX)/DET
        RYT=-RYX*OXT-RYY*OYT-RYZ*OZT
        RZX=(OYX*OZY-OYY*OZX)/DET
        RZY=(OXY*OZX-OZY*OXX)/DET
        RZZ=(OXX*OYY-OXY*OYX)/DET
        RZT=-RZX*OXT-RZY*OYT-RZZ*OZT
C MOVE HAND TO OBJECT REFERENCE FRAME
        X=RXX*XC+RXY*YC+RXZ*ZC+RXT
```

```
        Y=RYX*XC+RYY*YC+RYZ*ZC+RYT
        Z=RZX*XC+RZY*YC+RZZ*ZC+RZT
        XN=ABS(RXX*XY2+RXY*YY2+RXZ*ZY2)
        YN=ABS(RYX*XY2+RYY*YY2+RYZ*ZY2)
        ZN=ABS(RZX*XY2+RZY*YY2+RZZ*ZY2)
C DETERMINE IF HAND IF HOLDING OBJECT
        DO 119 I=1,NOP
C DETERMINE IF CENTER OF HAND IS WITHIN OBJECT
        IF(ABS(X-TCH(I,1)).GT.TCH(I,4))GOTO 119
        IF(ABS(Y-TCH(I,2)).GT.TCH(I,5))GOTO 119
        IF(ABS(Z-TCH(I,3)).GT.TCH(I,6))GOTO 119
C DETERMINE IF HAND IS CLOSED ENOUGH TO GRASP OBJECT
        IF(XN.NE.0)RS=TCH(I,4)/XN
        IF(YN.NE.0)RH=TCH(I,5)/YN
        IF(RH.LT.RS)RS=RH
        IF(ZN.NE.0)RH=TCH(I,6)/ZN
        IF(RH.LT.RS)RS=RH
        R=RS*SQRT(XN**2+YN**2+ZN**2)
        IF((1-R/1.9).GT.D)GOTO 119
C HOLD HAND OPEN TO WIDTH OF OBJECT
        DD=1-R/1.9
        NCENT=I
        GOTO 120
119     CONTINUE
        GOTO 122
120     IFO=1
C CALCULATE OBJECT TO DATA BASE TRANSFORMATION
        DET=XX2*YY2*ZZ2+XY2*YZ2*ZX2+XZ2*YX2*ZY2
     1  -ZX2*YY2*XZ2-ZY2*YZ2*XX2-ZZ2*YX2*XY2
        RXX=(YY2*ZZ2-ZY2*YZ2)/DET
        RXY=(ZY2*XZ2-XY2*ZZ2)/DET
        RXZ=(XY2*YZ2-YY2*XZ2)/DET
        RXT=-RXX*XT2-RXY*YT2-RXZ*ZT2
        RYX=(YZ2*ZX2-YX2*ZZ2)/DET
        RYY=(XX2*ZZ2-XZ2*ZX2)/DET
        RYZ=(YX2*XZ2-YZ2*XX2)/DET
        RYT=-RYX*XT2-RYY*YT2-RYZ*ZT2
        RZX=(YX2*ZY2-YY2*ZX2)/DET
        RZY=(XY2*ZX2-ZY2*XX2)/DET
        RZZ=(XX2*YY2-XY2*YX2)/DET
        RZT=-RZX*XT2-RZY*YT2-RZZ*ZT2
C MOVE OBJECT TO DATABASE
        HXX=RXX*OXX+RXY*OYX+RXZ*OZX
        HXY=RXX*OXY+RXY*OYY+RXZ*OZY
        HXZ=RXX*OXZ+RXY*OYZ+RXZ*OZZ
        HXT=RXX*OXT+RXY*OYT+RXZ*OZT+RXT
        HYX=RYX*OXX+RYY*OYX+RYZ*OZX
        HYY=RYX*OXY+RYY*OYY+RYZ*OZY
        HYZ=RYX*OXZ+RYY*OYZ+RYZ*OZZ
```

```
C       HYT=RYX*OXT+RYY*OYT+RYZ*OZT+RYT !UNCENTERED
C CENTER OBJECT
        HYT=-HYX*TCH(NCENT,1)-HYY*TCH(NCENT,2)
     1  -HYZ*TCH(NCENT,3)
        HZX=RZX*OXX+RZY*OYX+RZZ*OZX
        HZY=RZX*OXY+RZY*OYY+RZZ*OZY
        HZZ=RZX*OXZ+RZY*OYZ+RZZ*OZZ
        HZT=RZX*OXT+RZY*OYT+RZZ*OZT+RZT
C DETERMINE OBJECT POSITION
121     OXX=XX2*HXX+XY2*HYX+XZ2*HZX
        OXY=XX2*HXY+XY2*HYY+XZ2*HZY
        OXZ=XX2*HXZ+XY2*HYZ+XZ2*HZZ
        OXT=XX2*HXT+XY2*HYT+XZ2*HZT+XT2
        OYX=YX2*HXX+YY2*HYX+YZ2*HZX
        OYY=YX2*HXY+YY2*HYY+YZ2*HZY
        OYZ=YX2*HXZ+YY2*HYZ+YZ2*HZZ
        OYT=YX2*HXT+YY2*HYT+YZ2*HZT+YT2
        OZX=ZX2*HXX+ZY2*HYX+ZZ2*HZX
        OZY=ZX2*HXY+ZY2*HYY+ZZ2*HZY
        OZZ=ZX2*HXZ+ZY2*HYZ+ZZ2*HZZ
        OZT=ZX2*HXT+ZY2*HYT+ZZ2*HZT+ZT2
C SET VELOCITY OF OBJECT TO VELOCTY OF HAND
        OVX=(XC-OXO)/DELTIM
        OVY=(YC-OYO)/DELTIM
        OVZ=(ZC-OZO)/DELTIM
C SAVE HAND POSITION FOR VELOCITY DETERMINATION
        OXO=XC
        OYO=YC
        OZO=ZC
C SET FORCE FEEDBACK FLAG
        IFORCO=1
        D=DD
        GOTO 123
C IF HAND IS NOT HOLDING OBJECT
122     IFORCO=0
        IFO=0
C KEEP OBJECT WITHIN BOUNDARIES
        IF(ABS(OXT).GT.26.)OVX=-OVX*ELAST
        IF(ABS(OYT+24.).GT.24.)OVY=-OVY*ELAST
        IF(ABS(OZT+12.).GT.24.)OVZ=-OVZ*ELAST
C ACCELERATE/DECCELERATE OBJECT
        OXT=OXT+OVX*DELTIM
        OYT=OYT+OVY*DELTIM
        OZT=OZT+OVZ*DELTIM
        IF(OYT.GT.-48)OVY=OVY-G*DELTIM
        OVX=OVX*(1-DRAG)
        OVY=OVY*(1-DRAG)
        OVZ=OVZ*(1-DRAG)
C DETERMINE IF HAND IS TOUCHING SURFACE
```

```
123      IF(ISURF.NE.1)GOTO 125
         Y=YSURF(XT,ZT,P1,P2)
         BC=ABS(SIN(XT/5)*COS(ZT/10))
         DEP=Y-YT
         IF(YT.GT.Y)GOTO 124
C IF HAND IS TOUCHING SURFACE, CALCULATE RESISTIVE FORCE
         FX=0
         FY=DEP*BC
         FZ=0
         IFORCS=1
         GOTO 125
124      IFORCS=0
C SEND FORCE FEEDBACK IF TOUCHING SPHERE OR OBJECT
125      ITONGS=IFORCE.OR.IFORCO
         IARM=IFORCS
C TURN OFF FORCE FEEDBACK
         IF(ITONGS.EQ.0)CALL DOUT(24,"77)
         IF(IARM.EQ.0)ICTL=1
C TURN ON FORCE FEEDBACK
C APPLY FORCE TO TONGS
         IF(ITONGS.EQ.0)GOTO 126
         ICNTR=AB*3276.2*(D/SCL(7,1)-SCL(7,2))+
     1   (1-AB)*FLOAT(IA(8))
         CALL AOUT(11,ICNTR)
         CALL AOUT(4,-ICNTR)
         CALL DOUT(24,"177)
C APPLY FORCE TO ARM
126      IF(IARM.EQ.0)GOTO 128
         A1=FX*R11+FY*R12+FZ*R13
         A2=FX*R21+FY*R22+FZ*R23
         A3=FX*R31+FY*R32+FZ*R33
         A(5)=A(5)+A1/SCL(1,1)
         A(7)=A(7)+A2/SCL(2,1)
         A(6)=A(6)+(A1+A3)/SCL(3,1)
         DO 127 I=5,7
         IS(I)=A(I)*3276.2+F1*FLOAT(IS(I)-I3(I))
127      CONTINUE
         ICTL=0
         CALL AOUTSQ(11,17,IS)
C HAND CLOSURE TRANSFORMATIONS
C CALCULATE HAND POSITION
128      DO 129 I=2,5
         PNT(I,1)=ADAT(I,2)-1.9*D
         PNT(I,2)=ADAT(I,3)-.8*D
         PNT(I+10,1)=ADAT(I+10,2)-1.9*D
         PNT(I+10,2)=ADAT(I+10,3)-.8*D
         PNT(I+4,1)=ADAT(I+4,2)+1.9*D
         PNT(I+4,2)=ADAT(I+4,3)-.8*D
         PNT(I+14,1)=ADAT(I+14,2)+1.9*D
```

```
        PNT(I+14,2)=ADAT(I+14,3)-.8*D
129     CONTINUE
C ERASE PREVIOUS DISPLAY
        CALL LDPTRO(LOC)
C LOAD DISPLAY
C LOAD HAND INTO SUB 7
        CALL SUBDEF(7)
        DO 131 I=1,NHL
        MS=MM
        M=ACON(I,1)
        MM=ACON(I,2)
        IF(MS.EQ.M)GOTO 130
        IX=SF*ADAT(M,1)
        IY=SF*PNT(M,1)
        IZ=SF*PNT(M,2)
        CALL MOVI3(IX,IY,IZ)
130     IX=SF*ADAT(MM,1)
        IY=SF*PNT(MM,1)
        IZ=SF*PNT(MM,2)
        CALL DRWI3(IX,IY,IZ)
131     CONTINUE
        CALL SUBEND(7)
C LOAD SHOULDER
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX,XY,0.,0.,YX,YY,YZ,0.,ZX,ZY,ZZ,0.,SF)
        CALL SUBCAL(3)
C LOAD FOREARM
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX1,XY1,XZ1,XT1,YX1,YY1,YZ1,YT1,ZX1,ZY1,ZZ1,ZT1,SF)
        CALL SUBCAL(2)
C LOAD HAND
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX2,XY2,XZ2,XT2,YX2,YY2,YZ2,YT2,ZX2,ZY2,ZZ2,ZT2,SF)
        CALL SUBCAL(7)
C LOAD OBJECT
        IF(IOBJ.NE.1)GOTO 132
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  OXX,OXY,OXZ,OXT,OYX,OYY,OYZ,OYT,OZX,OZY,OZZ,OZT,SF)
        CALL SUBCAL(5)
C LOAD SHADOWS
132     IF(ISHAD.NE.1)GOTO 133
C LOAD SHOULDER SHADOW
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX,XY,0.,0.,0.,0.,0.,-48.75,ZX,ZY,ZZ,0.,SF)
        CALL SUBCAL(3)
C LOAD FOREARM SHADOW
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX1,XY1,XZ1,XT1,0.,0.,0.,-48.75,ZX1,ZY1,ZZ1,ZT1,SF)
        CALL SUBCAL(2)
```

```
C LOAD HAND SHADOW
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  XX2,XY2,XZ2,XT2,0.,0.,0.,-48.75,ZX2,ZY2,ZZ2,ZT2,SF)
        CALL SUBCAL(7)
C LOAD OBJECT SHADOW
        IF(IOBJ.NE.1)GOTO 133
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  OXX,OXY,OXZ,OXT,0.,0.,0.,-48.75,OZX,OZY,OZZ,OZT,SF)
        CALL SUBCAL(5)
133     CALL LDTRN3(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT)
C DISPLAY CENTER OF HAND
        IF(ITCHPT.NE.1)GOTO 134
        IX=SF*XC
        IY=SF*YC
        IZ=SF*ZC
        CALL MOVI3(IX,IY,IZ)
        CALL DRWI3(IX,IY,IZ)
C DISPLAY SHADOW
        IF(ISHAD.NE.1)GOTO 134
        IY=-1950
        CALL MOVI3(IX,IY,IZ)
        CALL DRWI3(IX,IY,IZ)
C DRAW WALLS
134     IF(IWALL.NE.1)GOTO 135
        CALL SUBCAL(1)
C LOAD PERMANANT DISPLAY
135     CALL SUBCAL(6)
C DISPLAY CROSS-SECTION OF SURFACE UNDER HAND
        IF(ISURF.NE.1)GOTO 137
        CALL SETINT(15)
        IY=SF*YSURF(-2000./SF,ZT,P1,P2)
        CALL MOVI3(-2000,IY,IFIX(SF*ZT))
        DO 136 I=-2000,2000,80
        X=I
        IY=SF*YSURF(X/SF,ZT,P1,P2)
        IF(DEP.GT.0)IY=FLOAT(IY)-DEP*SF*
     1  EXP(-.004*(1.-BC)*(X/SF-XT)**2.)
        CALL DRWI3(I,IY,IFIX(SF*ZT))
136     CONTINUE
C DISPLAY SPHERE
137     IF(ISPH.NE.1)GOTO 138
        IX=SF/S*(WXX*XO+WXZ*ZO)
        IY=SF/S*(WYX*XO+WYY*YO+WYZ*ZO)
        IZ=SF/S*(WZX*XO+WZY*YO+WZZ*ZO)
        CALL LDTRN3(S,0.,0.,WXT,0.,S,0.,WYT)
        CALL MOVI3(IX,IY,IZ)
        CALL SUBCAL(4)
C DISPLAY SPHERE SHADOW
        IF(ISHAD.NE.1)GOTO 138
```

```
        IX=SF*XO
        IY=SF*ZO
        IZ=-1950
        CALL LDTRN3(WXX,WXZ,WXY,WXT,WYX,WYZ,WYY,WYT)
        CALL MOVI3(IX,IY,IZ)
        CALL SUBCAL(4)
C DISPLAY COORDINATES
138     CALL LDTRN3(1.,0.,0.,0.,0.,1.,0.,0.)
        IF(ICOOR.NE.1)GOTO 139
        ENCODE(11,60,IBUF)XT
60      FORMAT('X=',F8.4)
        CALL MOVI3(-2000,1900,0)
        CALL CHAR(IBUF,11,2,0)
        ENCODE(11,61,IBUF)YT
61      FORMAT('Y=',F8.4)
        CALL MOVI3(-2000,1800,0)
        CALL CHAR(IBUF,11,2,0)
        ENCODE(11,62,IBUF)ZT
62      FORMAT('Z=',F8.4)
        CALL MOVI3(-2000,1700,0)
        CALL CHAR(IBUF,11,2,0)
        ENCODE(11,63,IBUF)BC
63      FORMAT('K=',F9.6)
        CALL MOVI3(-2000,1600,0)
        CALL CHAR(IBUF,11,2,0)
        ENCODE(11,64,IBUF)DELTIM
64      FORMAT('T=',F9.6)
        CALL MOVI3(-2000,1500,0)
        CALL CHAR(IBUF,11,2,0)
C DISPLAY PROXIMITY INDICATOR
139     IF(IPROX.NE.1)GOTO 140
        CALL MOVI3(0,1000,0)
        IF(PROX.GT.24)PROX=24.+(PROX-24.)/10.
        CALL RDRWI3(0,IFIX(40.*PROX),0)
        CALL MOVI3(-100,1000,0)
        CALL DRWI3(100,1000,0)
C SEND PICTURE TO DISPLAY BUFFER
140     CALL MGSEND
C READ KEYBOARD
        IF(IFF.NE.1)CALL QIO("10400,5,3,,IOSB,IPARAM,IDS)
        IFF=1
        CALL READEF(3,IUU)
        IF(IUU.NE.2)GOTO 114
C CLEAR ENTRY
        WRITE(5,999)"033,"114,"033,"101
999     FORMAT('+',4A)
        IFF=0
C DECODE INPUT
        IF(ICMD.EQ."070)DWA2=DWA2+.01
```

```
        IF(ICMD.EQ."062)DWA2=DWA2-.01
        IF(ICMD.EQ."064)DWA1=DWA1-.01
        IF(ICMD.EQ."066)DWA1=DWA1+.01
        IF(ICMD.EQ."075)DTY=DTY+40.
        IF(ICMD.EQ."134)DTX=DTX+40.
        IF(ICMD.EQ."047)DTY=DTY-40.
        IF(ICMD.EQ."133)DTX=DTX-40.
        IF(ICMD.NE."065)GOTO 141
        DWA1=0
        DWA2=0
        DS=0
        DTX=0
        DTY=0
141     IF(ICMD.EQ."071)DS=DS+.01
        IF(ICMD.EQ."063)DS=DS-.01
        IF(ICMD.EQ."132)STOP
        IF(ICMD.EQ."123)ISHAD=-ISHAD
        IF(ICMD.EQ."127)IWALL=-IWALL
        IF(ICMD.EQ."124)ITCHPT=-ITCHPT
        IF(ICMD.EQ."103)ICOOR=-ICOOR
        IF(ICMD.EQ."120)IPROX=-IPROX
        IF(ICMD.EQ."122)GOTO 100
        IF(ICMD.NE."104)GOTO 142
        IX=SF*XT
        IY=SF*YT
        IZ=SF*ZT
        CALL MODIFY(LINES)
        CALL DRWI3(IX,IY,IZ)
        LINES=LINES+2
        IF(ISHAD.EQ.1)GOTO 142
        CALL MODIFY(LSHADS)
        CALL DRWI3(IX,-1950,IZ)
        LSHADS=LSHADS+2
142     IF(ICMD.NE."115)GOTO 143
        IX=SF*XT
        IY=SF*YT
        IZ=SF*ZT
        CALL MODIFY(LINES)
        CALL MOVI3(IX,IY,IZ)
        LINES=LINES+2
        IF(ISHAD.EQ.1)GOTO 143
        CALL MODIFY(LSHADS)
        CALL MOVI3(IX,-1950,IZ)
        LSHADS=LSHADS+2
143     IF(ICMD.NE."105)GOTO 144
        LINES=LINES-2
        CALL MODIFY(LINES)
        CALL RMOVI3(0,0,0)
        IF(ISHAD.NE.1)GOTO 144
```

```
          LSHADS=LSHADS-2
          CALL MODIFY(LSHADS)
          CALL RMOVI3(0,0,0)
144       IF(ICMD.NE."067)GOTO 145
          WA1=1.57
          WA2=0.
145       IF(ICMD.NE."061)GOTO 146
          WA1=0
          WA2=0
146       IF(ICMD.NE."060)GOTO 114
          TX=0
          TY=0
          GOTO 114
          END
          FUNCTION YSURF(X,Z,P1,P2)
          YSURF=(SIN(.1256*Z)+COS(.02472*Z))*P1*X/(P2+X**2)-35.
          RETURN
          END
          SUBROUTINE DRAW(IPNTS,SCL)
          DO 200 I=1,IPNTS
          MS=MM
C READ CONNECTIVITY DATA
          READ(1,*,END=300)M,MM
C IF STARTING POINT WAS LAST LOCATION SKIP MOVE
          IF(M.EQ.MS)GOTO 100
C READ COORDINATES AND DRAW
          READ(2'M)X,Y,Z
          IX=SCL*X
          IY=SCL*Y
          IZ=SCL*Z
          CALL MOVI3(IX,IY,IZ)
100       READ(2'MM)X,Y,Z
          IX=SCL*X
          IY=SCL*Y
          IZ=SCL*Z
          CALL DRWI3(IX,IY,IZ)
200       CONTINUE
300       RETURN
          END
          SUBROUTINE LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
       1  XX,XY,XZ,XT,YX,YY,YZ,YT,ZX,ZY,ZZ,ZT,SF)
          CALL LDTRN3(WXX*XX+WXY*YX+WXZ*ZX,WXX*XY+WXY*YY+WXZ*ZY,
       1  WXX*XZ+WXY*YZ+WXZ*ZZ,SF*(WXX*XT+WXY*YT+WXZ*ZT)+WXT,
       1  WYX*XX+WYY*YX+WYZ*ZX,WYX*XY+WYY*YY+WYZ*ZY,WYX*XZ
       1  +WYY*YZ+WYZ*ZZ,SF*(WYX*XT+WYY*YT+WYZ*ZT)+WYT)
          RETURN
          END
```

APPENDIX VI

SUBMARINE SIMULATION SOFTWARE

```
C INITIALIZE PROGRAM
        DIMENSION IA(16),A(7),IPARAM(6),IOSB(2),IBUF(12)
        BYTE ACON(34,2)
        COMMON /DMABUF/IDUM(3610),ADAT(22,3),PNT(22,2),
     1  TCH(3,6),SCL(7,2),II(7),I1(7),I2(7),I3(7),IS(7),F1,
     1  IT1,IT2,ICTL,VEC(18),ALNGTH,AMP,INDX
        DATA OXX,OXY,OXZ/1.,0.,0./
        DATA OYX,OYY,OYZ/0.,1.,0./
        DATA OZX,OZY,OZZ/0.,0.,1./
        ICTL=0
100     WRITE(5,50)
50      FORMAT(' ENTER FUNCTION'/' 1=SPHERE'/' 2=OBJECT'/
     1  ' 4=SURFACE'/' 8=PATH'/' 16=NPATH')
        READ(5,*)IFUNC
        ISPH=0
        IOBJ=0
        ISURF=0
        IPATH=0
        INPATH=0
        IF((IFUNC.AND."1).NE.0)ISPH=1
        IF((IFUNC.AND."2).NE.0)IOBJ=1
        IF((IFUNC.AND."4).NE.0)ISURF=1
        IF((IFUNC.AND."10).NE.0)IPATH=1
        IF((IFUNC.AND."20).NE.0)INPATH=1
C INITIALIZE THE MEGATEK
        CALL MGINIT
        CALL SETINT(13)
C INITALIZE THE A/D
        CALL ANINIT
        CALL AINSQ(16,29,IA)
        CALL AOUTSQ(4,17,IA)
        WRITE(5,51)
51      FORMAT(' FEEDFORWARD ACTIVE AND PASSIVE CYCLE TIMES
     1  (.7,30,150)')
        READ(5,*)F1,IT1,IT2
        ICTL=1
        CALL DOUT(24,"77)
C INITIALIZE THE KEYBOARD MONITER ROUTINE
        CALL GETADR(IPARAM(1),ICMD)
        IPARAM(2)=1
C SET FUNCTION FLAGS
        ISHAD=-1          !SHADOW
        IWALL=-1          !WALLS
        ITCHPT=-1         !TOUCHPOUNTS
        IPROX=-1          !PROXIMITY INDICATOR
```

```
C SET POSITION OF ARM PARTS
        ZHC=-5.          !CENTER OF HAND
        ZHT=-5.6         !END OF HAND
        ZFT=-40.3        !END OF FOREARM
        ZST=-18.4        !END OF SHOULDER
        YSUBT=-7.5       !CENTER OF SUBMARINE ROTATION
        ZSUBT=-15.       !CENTER OF SUBMARINE ROTATION
        SF=40.           !DISPLAY SCALE
        NSUBL=40         !# LINES IN SUBMARINE
        NHL=34           !# LINES IN HAND
        NSL=16           !# LINES IN SHOULDER
        NFL=20           !# LINES IN FORARM
C SET POSITION OF OBJECTS
        XO=0
        YO=-40
        ZO=-24
        OXT=-10
        OYT=-40
        OZT=-20
        IFLG=0
        IFLGO=0
        IFLGS=0
C SET INITIAL VELOCITY TO ZERO
        VX=0.
        VY=0.
        VZ=0.
        OVX=0
        OVY=0
        OVZ=0
        TIM=SECNDS(0.)
C SET VIEWPOINT AND ZOOM
        WRITE(5,45)
45      FORMAT(' ENTER DISPLAY SCALE FACTOR (<1)')
        READ(5,*)S
        SF=SF*S
        TX=0.
        TY=0.
        S=1
        WA1=0
        WA2=0
        DWA1=0
        DWA2=0
        DTX=0
        DTY=0
        DS=0
C READ SCALING FACTORS FOR A/D OUTPUT OF ANGLES
        OPEN(UNIT=4,NAME='DL1:[200,125]SCALE.DAT',TYPE='OLD')
        READ(4,*)((SCL(I,J),J=1,2),I=1,7)
        CLOSE(UNIT=4,DISPOSE='SAVE')
```

```
C READ POINT DATA
        OPEN(UNIT=4,NAME='DL1:[200,125]ARMDAT.DAT',TYPE='OLD')
        READ(4,*)((ADAT(I,J),J=1,3),I=1,22)
        CLOSE(UNIT=4,DISPOSE='SAVE')
C READ CONNECTIVITY DATA
        OPEN(UNIT=4,NAME='DL1:[200,125]ARMCON.DAT',TYPE='OLD')
        READ(4,*)((ACON(I,J),J=1,2),I=1,NHL)
        CLOSE(UNIT=4,DISPOSE='SAVE')
C READ TOUCHING CONDITIONS
        OPEN(UNIT=4,NAME='DL1:[200,125]OBJTCH.DAT',TYPE='OLD')
        DO 204 I=1,5
        NOP=I-1
        READ(4,*,END=205)(TCH(I,J),J=1,6)
204     CONTINUE
205     CLOSE(UNIT=4,DISPOSE='SAVE')
C READ PARAMETERS OF SURFACE
206     IF(ISURF.NE.1)GOTO 207
        WRITE(5,1)
1       FORMAT(' ENTER PARAMETERS P1,P2 OF SURFACE (200,300)')
        READ(5,*)P1,P2
C READ DESIRED DECCELLERATION, GRAVITY, ELASTICITY
207     IF((IOBJ.NE.1).AND.(ISPH.NE.1))GOTO 208
        WRITE(5,2)
2       FORMAT(' DECCELERATION, GRAVITY, STIFFNESS,
     1  ELASTICITY (0<<1)')
        READ(5,*)DRAG,G,AB,ELAST
C PRINT DISPLAY CONTROL INSTRUCTIONS
208     WRITE(5,3)
3       FORMAT(' TO ROTATE TYPE:')
        WRITE(5,4)
4       FORMAT(' 8=UP  2=DOWN  6=RIGHT  4=LEFT  5=STOP
     1  1=FRONT  7=SIDE')
        WRITE(5,5)
5       FORMAT(' TO TRANSLATE TYPE:')
        WRITE(5,6)"047  !QUOTE
6       FORMAT(' "="=UP  ',A,'=DOWN  [=RIGHT  ®=LEFT
     1  0=CENTER')
        WRITE(5,7)
7       FORMAT(' TO ZOOM TYPE:')
        WRITE(5,8)
8       FORMAT(' 9=ZOOM UP  3=ZOOM DOWN')
        WRITE(5,9)
9       FORMAT(' TO DRAW LINES TYPE:')
        WRITE(5,10)
10      FORMAT(' M=MOVE  D=DRAW  E=ERASE LAST LINE')
        WRITE(5,11)
11      FORMAT(' TYPE "S" FOR SHADOW, "T" FOR TOUCHPOINTS')
        WRITE(5,12)
12      FORMAT(' TYPE "Z" TO EXIT, "R" TO REPEAT')
```

```
C LOAD SUBPICTURES
C LOAD SUBMARINE IN SUB 1
        OPEN(UNIT=1,NAME='DL1:[200,125]SUBCON.DAT',TYPE='OLD')
        OPEN(UNIT=2,NAME='DL1:[200,125]SUBDAT.DAF',TYPE='OLD'
     1  ,ACCESS='DIRECT',RECORDSIZE=3)
        CALL SUBDEF(1)
        CALL DRAW(100,SF)
        CALL SUBEND(1)
        CLOSE(UNIT=1,DISPOSE='SAVE')
        CLOSE(UNIT=2,DISPOSE='SAVE')
        OPEN(UNIT=1,NAME='DL1:[200,125]ARMCON.DAT',TYPE='OLD')
        OPEN(UNIT=2,NAME='DL1:[200,125]ARMDAT.DAF',TYPE='OLD'
     1  ,ACCESS='DIRECT',RECORDSIZE=3)
C SPOOL THROUGH FILE TO FOREARM DATA
        DO 933 I=1,NHL
        READ(1,*)M,MM
933     CONTINUE
C LOAD FOREARM IN SUB 2
        CALL SUBDEF(2)
        CALL DRAW(NFL,SF)
        CALL SUBEND(2)
C LOAD SHOULDER IN SUB 3
        CALL SUBDEF(3)
        CALL DRAW(NSL,SF)
        CALL SUBEND(3)
        CLOSE(UNIT=1,DISPOSE='SAVE')
        CLOSE(UNIT=2,DISPOSE='SAVE')
        I=SF*.525
        J=SF*.225
C LOAD SPHERE IN SUB 4
        CALL SUBDEF(4)
        CALL RMOVI3(0,IFIX(SF*.75),0)
        CALL RDRWI3(I,-J,0)
        CALL RDRWI3(J,-I,0)
        CALL RDRWI3(-J,-I,0)
        CALL RDRWI3(-I,-J,0)
        CALL RDRWI3(-I,J,0)
        CALL RDRWI3(-J,I,0)
        CALL RDRWI3(J,I,0)
        CALL RDRWI3(I,J,0)
        CALL SUBEND(4)
C LOAD OBJECT INTO SUB 5
        OPEN(UNIT=1,NAME='DL1:[200,125]OBJCON.DAT',TYPE='OLD')
        OPEN(UNIT=2,NAME='DL1:[200,125]OBJDAT.DAF',TYPE='OLD'
     1  ,ACCESS='DIRECT',RECORDSIZE=3)
        CALL SUBDEF(5)
        CALL DRAW(100,SF)
        CALL SUBEND(5)
        CLOSE(UNIT=1,DISPOSE='SAVE')
```

```
        CLOSE(UNIT=2,DISPOSE='SAVE')
C LOAD PERMANANT DISPLAY INTO SUB 6
        CALL SUBDEF(6)
        IF(ISURF.NE.1)GOTO 759
        CALL SETINT(10)
        DO 106 I=-2000,2000,200
        Z=I
        X=-2000.
        IY=SF*YSURF(X/SF,Z/SF,P1,P2)
        CALL MOVI3(-2000,IY,I)
        DO 105 J=-2000,2000,200
        X=J
        IY=SF*YSURF(X/SF,Z/SF,P1,P2)
        CALL DRWI3(J,IY,I)
105     CONTINUE
        CALL MGSEND
106     CONTINUE
        DO 108 I=-2000,2000,200
        X=I
        Z=-2000.
        IY=SF*YSURF(X/SF,Z/SF,P1,P2)
        CALL MOVI3(I,IY,-2000)
        DO 107 J=-2000,2000,200
        Z=J
        IY=SF*YSURF(X/SF,Z/SF,P1,P2)
        CALL DRWI3(I,IY,J)
107     CONTINUE
        CALL MGSEND
108     CONTINUE
        CALL SETINT(13)
C LOAD PATH FOLLOWING PATHS
759     IF(INPATH.NE.1)GOTO 561
        CALL MOVI3(-2000,-1000,0)
        DO 560 I=-2000,2000,10
        PX=FLOAT(I)/SF/12.
        CALL NPATH(PX,Z,DL,DR,HEIGT,X1,X3,PY,INDX)
        CALL DRWI3(I,-IFIX(12.*PY*SF),0)
560     CONTINUE
561     IF(IPATH.NE.1)GOTO 109
        CALL MOVI3(-2000,-1000,0)
        DO 562 I=-2000,2000,10
        PX=FLOAT(I)/SF/12.
        CALL PATH(AMP,ALNGTH,PX,PEL)
        CALL DRWI3(I,-IFIX(12.*PEL*SF)-1300,0)
562     CONTINUE
109     CALL NPOINT(LINES)
C SAVE MEMORY SPACE FOR INPUTTING LINES
        DO 110 I=1,40
        CALL MOVI3(0,0,0)
```

```
110     CONTINUE
        CALL NPOINT(LSHADS)
        DO 111 I=1,40
        CALL MOVI3(0,0,0)
111     CONTINUE
        CALL SUBEND(6)
C LOAD INITIAL ARM POSITION
        DO 834 I=1,22
        PNT(I,1)=ADAT(I,2)
        PNT(I,2)=ADAT(I,3)
834     CONTINUE
        CALL NPOINT(LOC)
C READ ARM POSITION FROM A/D CONVERTER AND CONVERT TO VOLTAGE
112     CALL AINSQ(23,29,IS)
        DO 113 I=1,7
        I3(I)=IS(I)
        A(I)=FLOAT(IS(I))/3276.2
113     CONTINUE
C INCREMENT VIEWPOINT
        WA1=WA1+DWA1
        WA2=WA2+DWA2
        S=S+DS
        TX=TX+DTX
        TY=TY+DTY
C SCALE A/D OUTPUT AND CALCULATE SINES & COSINES
914     AH=SCL(1,1)*(A(5)+SCL(1,2))
        S1=SIN(AH)
        C1=COS(AH)
        AH=SCL(2,1)*(A(7)+SCL(2,2))
        S2=SIN(AH)
        C2=COS(AH)
        AH=SCL(3,1)*(A(6)+SCL(3,2))-SCL(1,1)*(A(5)+SCL(1,2))
        S3=SIN(AH)
        C3=COS(AH)
        AH=SCL(4,1)*(A(2)+SCL(4,2))
        S4=SIN(AH)
        C4=COS(AH)
        AH=SCL(5,1)*((A(3)-A(4))/2.+SCL(5,2))
        S5=SIN(AH)
        C5=COS(AH)
        AH=SCL(6,1)*((A(3)+A(4))/2.+SCL(6,2))
        S6=SIN(AH)
        C6=COS(AH)
        D=SCL(7,1)*(A(1)+SCL(7,2))
        SV1=SIN(VEC(7))
        CV1=COS(VEC(7))
        SV2=SIN(VEC(9))
        CV2=COS(VEC(9))
        SV3=SIN(VEC(8))
```

```
      CV3=COS(VEC(8))
      WS1=SIN(WA1)
      WC1=COS(WA1)
      WS2=SIN(WA2)
      WC2=COS(WA2)
      HCR=ZHC-.8*D
      HTR=ZHT-.8*D
C CALCALATE TRANSFORMATIONS
C CALCULATE VIEWPOINT TRANSFORMATIONS
      WXX=WC1*S
      WXY=0
      WXZ=WS1*S
      WXT=TX*S/SF
      WYX=WS1*WS2*S
      WYY=WC2*S
      WYZ=-WC1*WS2*S
      WYT=TY*S/SF
C CALCULATE SUBMARINE TRANSFORMATIONS
      SXX=CV1*CV2+SV1*SV2*SV3
      SXY=-SV1*CV2+CV1*SV2*SV3
      SXZ=SV2*CV3
      SXT=VEC(11)*12.
      SYX=SV1*CV3
      SYY=CV1*CV3
      SYZ=-SV3
      SYT=VEC(12)*12.+YSUBT
      SZX=-CV1*SV2+SV1*CV2*SV3
      SZY=SV1*SV2+CV1*CV2*SV3
      SZZ=CV2*CV3
      SZT=VEC(10)*12.+ZSUBT
C CALCULATE dX/dA1
      YY=S1*S2*S4-S1*C2*C3*C4-C1*S3*C4
      YZ=S1*C2*S3-C1*C3
      YT=-C1*ZST
      ZY=-C1*S2*S4+C1*C2*C3*C4-S1*S3*C4
      ZZ=-C1*C2*S3-S1*C3
      ZT=-S1*ZST
      Y=(-YY*S5+YZ*C5)*HTR+YZ*ZFT+YT
      Z=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT+ZT
      DXD1=SXY*Y+SXZ*Z
      DYD1=SYY*Y+SYZ*Z
      DZD1=SZY*Y+SZZ*Z
C CALCULATE dX/dA2
      XY=S2*S4-C2*C3*C4
      XZ=C2*S3
      YY=-C1*C2*S4-C1*S2*C3*C4
      YZ=C1*S2*S3
      ZY=-S1*C2*S4-S1*S2*C3*C4
      ZZ=S1*S2*S3
```

```
      X=(-XY*S5+XZ*C5)*HTR+XZ*ZFT
      Y=(-YY*S5+YZ*C5)*HTR+YZ*ZFT
      Z=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT
      DXD2=SXX*X+SXY*Y+SXZ*Z
      DYD2=SYX*X+SYY*Y+SYZ*Z
      DZD2=SZX*X+SZY*Y+SZZ*Z
C CALCULATE dX/dA3
      XY=S2*S3*C4
      XZ=S2*C3
      YY=-C1*C2*S3*C4-S1*C3*C4
      YZ=-C1*C2*C3+S1*S3
      ZY=-S1*C2*S3*C4+C1*C3*C4
      ZZ=-S1*C2*C3-C1*S3
      X=(-XY*S5+XZ*C5)*HTR+XZ*ZFT
      Y=(-YY*S5+YZ*C5)*HTR+YZ*ZFT
      Z=(-ZY*S5+ZZ*C5)*HTR+ZZ*ZFT
      DXD3=SXX*X+SXY*Y+SXZ*Z
      DYD3=SYX*X+SYY*Y+SYZ*Z
      DZD3=SZX*X+SZY*Y+SZZ*Z
C CALCULATE INVERSE
      DET=DXD1*DYD2*DZD3+DXD2*DYD3*DZD1+DXD3*DYD1*DZD2
     1  -DZD1*DYD2*DXD3-DYD1*DXD2*DZD3-DXD1*DYD3*DZD2
      R11=(DYD2*DZD3-DYD3*DZD2)/DET
      R12=(DZD2*DXD3-DXD2*DZD3)/DET
      R13=(DXD2*DYD3-DYD2*DXD3)/DET
      R21=(DYD3*DZD1-DYD1*DZD3)/DET
      R22=(DXD1*DZD3-DXD3*DZD1)/DET
      R23=(DYD1*DXD3-DXD1*DYD3)/DET
      R31=(DYD1*DZD2-DYD2*DZD1)/DET
      R32=(DXD2*DZD1-DZD2*DXD1)/DET
      R33=(DXD1*DYD2-DXD2*DYD1)/DET
C SHOULDER TRANSFORMATIONS
      XX=C2
      XY=-S2
      YX=C1*S2
      YY=C1*C2
      YZ=-S1
      ZX=S1*S2
      ZY=S1*C2
      ZZ=C1
C FOREARM TRANSFORMATIONS
      XX1=XX*C4+XY*C3*S4
      XY1=-XX*S4+XY*C3*C4
      XZ1=-XY*S3
      YX1=YX*C4+YY*C3*S4+YZ*S3*S4
      YY1=-YX*S4+YY*C3*C4+YZ*S3*C4
      YZ1=-YY*S3+YZ*C3
      YT1=YZ*ZST
      ZX1=ZX*C4+ZY*C3*S4+ZZ*S3*S4
```

```
      ZY1=-ZX*S4+ZY*C3*C4+ZZ*S3*C4
      ZZ1=-ZY*S3+ZZ*C3
      ZT1=ZZ*ZST
C HAND TRANSFORMATIONS
      XX2=XX1*C6+XY1*C5*S6+XZ1*S5*S6
      XY2=-XX1*S6+XY1*C5*C6+XZ1*S5*C6
      XZ2=-XY1*S5+XZ1*C5
      XT2=XZ1*ZFT
      YX2=YX1*C6+YY1*C5*S6+YZ1*S5*S6
      YY2=-YX1*S6+YY1*C5*C6+YZ1*S5*C6
      YZ2=-YY1*S5+YZ1*C5
      YT2=YZ1*ZFT+YT1
      ZX2=ZX1*C6+ZY1*C5*S6+ZZ1*S5*S6
      ZY2=-ZX1*S6+ZY1*C5*C6+ZZ1*S5*C6
      ZZ2=-ZY1*S5+ZZ1*C5
      ZT2=ZZ1*ZFT+ZT1
C COMBINED WORLD-SUB TRANSFORMATIONS
      WSXX=WXX*SXX+WXY*SYX+WXZ*SZX
      WSXY=WXX*SXY+WXY*SYY+WXZ*SZY
      WSXZ=WXX*SXZ+WXY*SYZ+WXZ*SZZ
      WSXT=WXX*SXT+WXY*SYT+WXZ*SZT+WXT
      WSYX=WYX*SXX+WYY*SYX+WYZ*SZX
      WSYY=WYX*SXY+WYY*SYY+WYZ*SZY
      WSYZ=WYX*SXZ+WYY*SYZ+WYZ*SZZ
      WSYT=WYX*SXT+WYY*SYT+WYZ*SZT+WYT
C WORLD-SUB TRANSFORMATION OF SHADOW
      SWXX=WXX*SXX+WXZ*SZX
      SWXY=WXX*SXY+WXZ*SZY
      SWXZ=WXX*SXZ+WXZ*SZZ
      SWXT=WXX*SXT-WXY*48.75+WXZ*SZT+WXT
      SWYX=WYX*SXX+WYZ*SZX
      SWYY=WYX*SXY+WYZ*SZY
      SWYZ=WYX*SXZ+WYZ*SZZ
      SWYT=WYX*SXT-WYY*48.75+WYZ*SZT+WYT
C DETERMINE POINT MIDWAY BETWEEN JAWS OF HAND
C FIND ELAPSED TIME
      OLDTIM=TIM
      TIM=SECNDS(0.)
      DELTIM=TIM-OLDTIM
C DETERMINE POINT MIDWAY BETWEEN JAWS OF HAND
      X=XZ2*HCR+XT2
      Y=YZ2*HCR+YT2
      Z=ZZ2*HCR+ZT2
      XC=SXX*X+SXY*Y+SXZ*Z+SXT
      YC=SYX*X+SYY*Y+SYZ*Z+SYT
      ZC=SZX*X+SZY*Y+SZZ*Z+SZT
      X=XZ2*HTR+XT2
      Y=YZ2*HTR+YT2
      Z=ZZ2*HTR+ZT2
```

```
        XT=SXX*X+SXY*Y+SXZ*Z+SXT
        YT=SYX*X+SYY*Y+SYZ*Z+SYT
        ZT=SZX*X+SZY*Y+SZZ*Z+SZT
C DETERMINE IF HAND IS HOLDING SPHERE
        IF(ISPH.NE.1)GOTO 121
C SKIP IF OBJECT GRIPPED LAST CYCLE AND JAWS STILL CLOSED
        IF((IFS.EQ.1).AND.(D.GE..6))GOTO 333
        IF((ABS(XC-XO).GT.1.).OR.(ABS(YC-YO).GT.1.).OR.
     1  (ABS(ZC-ZO).GT.1.).OR.(D.LT..6))GOTO 117
C IF HAND IS HOLDING SPHERE
C SET VELOCITY OF SPHERE TO VELOCTY OF HAND
        IFS=1
333     VX=(XC-XO)/DELTIM
        VY=(YC-YO)/DELTIM
        VZ=(ZC-ZO)/DELTIM
C SET CENTER OF SPHERE TO POSITION OF HAND
        XO=XC
        YO=YC
        ZO=ZC
C HOLD HAND OPEN AROUND SPHERE
        D=.6
C SET FORCE FEEDBACK FLAG
        IFORCE=1
        GOTO 121
C IF HAND IS NOT HOLDING SPHERE
117     IFORCE=0
        IFS=0
C KEEP SPHERE WITHIN BOUNDARIES
        IF(ABS(XO).GT.26.)VX=-VX*ELAST
        IF(ABS(YO+24.).GT.24.)VY=-VY*ELAST
        IF(ABS(ZO+12.).GT.24.)VZ=-VZ*ELAST
C ACCELERATE/DECCELERATE SPHERE
        XO=XO+VX*DELTIM
        YO=YO+VY*DELTIM
        ZO=ZO+VZ*DELTIM
        IF(YO.GT.-48)VY=VY-G*DELTIM
        VX=VX*(1-DRAG)
        VY=VY*(1-DRAG)
        VZ=VZ*(1-DRAG)
        PROX=SQRT((XO-XC)**2+(YO-YC)**2+(ZO-ZC)**2)
C DETERMINE IF HAND IS HOLDING OBJECT
121     IF(IOBJ.NE.1)GOTO 321
        IF((IFO.EQ.1).AND.(DD.LE.D))GOTO 315
C FIND HAND REFERENCE TO OBJECT REFERENCE TRANSFORMATION
        DET=OXX*OYY*OZZ+OXY*OYZ*OZX+OXZ*OYX*OZY
     1  -OZX*OYY*OXZ-OZY*OYZ*OXX-OZZ*OYX*OXY
        RXX=(OYY*OZZ-OZY*OYZ)/DET
        RXY=(OZY*OXZ-OXY*OZZ)/DET
        RXZ=(OXY*OYZ-OYY*OXZ)/DET
```

```
      RXT=-RXX*OXT-RXY*OYT-RXZ*OZT
      RYX=(OYZ*OZX-OYX*OZZ)/DET
      RYY=(OXX*OZZ-OXZ*OZX)/DET
      RYZ=(OYX*OXZ-OYZ*OXX)/DET
      RYT=-RYX*OXT-RYY*OYT-RYZ*OZT
      RZX=(OYX*OZY-OYY*OZX)/DET
      RZY=(OXY*OZX-OZY*OXX)/DET
      RZZ=(OXX*OYY-OXY*OYX)/DET
      RZT=-RZX*OXT-RZY*OYT-RZZ*OZT
C MOVE HAND TO OBJECT REFERENCE FRAME
      X=RXX*XC+RXY*YC+RXZ*ZC+RXT
      Y=RYX*XC+RYY*YC+RYZ*ZC+RYT
      Z=RZX*XC+RZY*YC+RZZ*ZC+RZT
      XN=ABS(RXX*XY2+RXY*YY2+RXZ*ZY2)
      YN=ABS(RYX*XY2+RYY*YY2+RYZ*ZY2)
      ZN=ABS(RZX*XY2+RZY*YY2+RZZ*ZY2)
C DETERMINE IF HAND IF HOLDING OBJECT
      DO 323 I=1,NOP
C DETERMINE IF CENTER OF HAND IS WITHIN OBJECT
      IF(ABS(X-TCH(I,1)).GT.TCH(I,4))GOTO 323
      IF(ABS(Y-TCH(I,2)).GT.TCH(I,5))GOTO 323
      IF(ABS(Z-TCH(I,3)).GT.TCH(I,6))GOTO 323
C DETERMINE IF HAND IS CLOSED ENOUGH TO GRASP OBJECT
      IF(XN.NE.0)RS=TCH(I,4)/XN
      IF(YN.NE.0)RH=TCH(I,5)/YN
      IF(RH.LT.RS)RS=RH
      IF(ZN.NE.0)RH=TCH(I,6)/ZN
      IF(RH.LT.RS)RS=RH
      R=RS*SQRT(XN**2+YN**2+ZN**2)
      IF((1-R/1.9).GT.D)GOTO 323
C HOLD HAND OPEN TO WIDTH OF OBJECT
      DD=1-R/1.9
      NCENT=I
      GOTO 320
323   CONTINUE
      GOTO 317
320   IFO=1
C CALCULATE OBJECT TO DATA BASE TRANSFORMATION
      HXX=SXX*XX2+SXY*YX2+SXZ*ZX2
      HXY=SXX*XY2+SXY*YY2+SXZ*ZY2
      HXZ=SXX*XZ2+SXY*YZ2+SXZ*ZZ2
      HXT=SXX*XT2+SXY*YT2+SXZ*ZT2+SXT
      HYX=SYX*XX2+SYY*YX2+SYZ*ZX2
      HYY=SYX*XY2+SYY*YY2+SYZ*ZY2
      HYZ=SYX*XZ2+SYY*YZ2+SYZ*ZZ2
      HYT=SYX*XT2+SYY*YT2+SYZ*ZT2+SYT
      HZX=SZX*XX2+SZY*YX2+SZZ*ZX2
      HZY=SZX*XY2+SZY*YY2+SZZ*ZY2
      HZZ=SZX*XZ2+SZY*YZ2+SZZ*ZZ2
```

```
      HZT=SZX*XT2+SZY*YT2+SZZ*ZT2+SZT
      DET=HXX*HYY*HZZ+HXY*HYZ*HZX+HXZ*HYX*HZY
     1  -HZX*HYY*HXZ-HZY*HYZ*HXX-HZZ*HYX*HXY
      RXX=(HYY*HZZ-HZY*HYZ)/DET
      RXY=(HZY*HXZ-HXY*HZZ)/DET
      RXZ=(HXY*HYZ-HYY*HXZ)/DET
      RXT=-RXX*HXT-RXY*HYT-RXZ*HZT
      RYX=(HYZ*HZX-HYX*HZZ)/DET
      RYY=(HXX*HZZ-HXZ*HZX)/DET
      RYZ=(HYX*HXZ-HYZ*HXX)/DET
      RYT=-RYX*HXT-RYY*HYT-RYZ*HZT
      RZX=(HYX*HZY-HYY*HZX)/DET
      RZY=(HXY*HZX-HZY*HXX)/DET
      RZZ=(HXX*HYY-HXY*HYX)/DET
      RZT=-RZX*HXT-RZY*HYT-RZZ*HZT
C MOVE OBJECT TO DATABASE
      HXX=RXX*OXX+RXY*OYX+RXZ*OZX
      HXY=RXX*OXY+RXY*OYY+RXZ*OZY
      HXZ=RXX*OXZ+RXY*OYZ+RXZ*OZZ
      HXT=RXX*OXT+RXY*OYT+RXZ*OZT+RXT
      HYX=RYX*OXX+RYY*OYX+RYZ*OZX
      HYY=RYX*OXY+RYY*OYY+RYZ*OZY
      HYZ=RYX*OXZ+RYY*OYZ+RYZ*OZZ
C     HYT=RYX*OXT+RYY*OYT+RYZ*OZT+RYT !UNCENTERED
C CENTER OBJECT WITHIN GRIPS
      HYT=-HYX*TCH(NCENT,1)-HYY*TCH(NCENT,2)
     1  -HYZ*TCH(NCENT,3)
      HZX=RZX*OXX+RZY*OYX+RZZ*OZX
      HZY=RZX*OXY+RZY*OYY+RZZ*OZY
      HZZ=RZX*OXZ+RZY*OYZ+RZZ*OZZ
      HZT=RZX*OXT+RZY*OYT+RZZ*OZT+RZT
C DETERMINE OBJECT POSITION
315   RXX=SXX*XX2+SXY*YX2+SXZ*ZX2
      RXY=SXX*XY2+SXY*YY2+SXZ*ZY2
      RXZ=SXX*XZ2+SXY*YZ2+SXZ*ZZ2
      RXT=SXX*XT2+SXY*YT2+SXZ*ZT2+SXT
      RYX=SYX*XX2+SYY*YX2+SYZ*ZX2
      RYY=SYX*XY2+SYY*YY2+SYZ*ZY2
      RYZ=SYX*XZ2+SYY*YZ2+SYZ*ZZ2
      RYT=SYX*XT2+SYY*YT2+SYZ*ZT2+SYT
      RZX=SZX*XX2+SZY*YX2+SZZ*ZX2
      RZY=SZX*XY2+SZY*YY2+SZZ*ZY2
      RZZ=SZX*XZ2+SZY*YZ2+SZZ*ZZ2
      RZT=SZX*XT2+SZY*YT2+SZZ*ZT2+SZT
      OXX=RXX*HXX+RXY*HYX+RXZ*HZX
      OXY=RXX*HXY+RXY*HYY+RXZ*HZY
      OXZ=RXX*HXZ+RXY*HYZ+RXZ*HZZ
      OXT=RXX*HXT+RXY*HYT+RXZ*HZT+RXT
      OYX=RYX*HXX+RYY*HYX+RYZ*HZX
```

```
          OYY=RYX*HXY+RYY*HYY+RYZ*HZY
          OYZ=RYX*HXZ+RYY*HYZ+RYZ*HZZ
          OYT=RYX*HXT+RYY*HYT+RYZ*HZT+RYT
          OZX=RZX*HXX+RZY*HYX+RZZ*HZX
          OZY=RZX*HXY+RZY*HYY+RZZ*HZY
          OZZ=RZX*HXZ+RZY*HYZ+RZZ*HZZ
          OZT=RZX*HXT+RZY*HYT+RZZ*HZT+RZT
C SET VELOCITY OF OBJECT TO VELOCTY OF HAND
          OVX=(XC-OXO)/DELTIM
          OVY=(YC-OYO)/DELTIM
          OVZ=(ZC-OZO)/DELTIM
C SAVE  HAND POSITION FOR VELOCITY DETERMINATION
          OXO=XC
          OYO=YC
          OZO=ZC
C SET FORCE FEEDBACK FLAG
          IFORCO=1
          D=DD
          GOTO 321
C IF HAND IS NOT HOLDING OBJECT
317       IFORCO=0
          IFO=0
C KEEP OBJECT WITHIN BOUNDARIES
          IF(ABS(OXT).GT.26.)OVX=-OVX*ELAST
          IF(ABS(OYT+24.).GT.24.)OVY=-OVY*ELAST
          IF(ABS(OZT+12.).GT.24.)OVZ=-OVZ*ELAST
C ACCELERATE/DECCELERATE OBJECT
          OXT=OXT+OVX*DELTIM
          OYT=OYT+OVY*DELTIM
          OZT=OZT+OVZ*DELTIM
          IF(OYT.GT.-48)OVY=OVY-G*DELTIM
          OVX=OVX*(1-DRAG)
          OVY=OVY*(1-DRAG)
          OVZ=OVZ*(1-DRAG)
C DETERMINE IF HAND IS TOUCHING SURFACE
321       IF(ISURF.NE.1)GOTO 123
          Y=YSURF(XT,ZT,P1,P2)
          BC=ABS(SIN(XT/5)*COS(ZT/10))
          DEP=Y-YT
          IF(YT.GT.Y)GOTO 413
C IF HAND IS TOUCHING SURFACE, CALCULATE RESISTIVE FORCE
          FX=0
          FY=DEP*BC
          FZ=0
          IFORCS=1
          GOTO 123
413       IFORCS=0
C SEND FORCE FEEDBACK
123       ITONGS=IFORCE.OR.IFORCO
```

```
        IARM=IFORCS
C TURN OFF FORCE FEEDBACK
        IF(ITONGS.EQ.0)CALL DOUT(24,"77)
        IF(IARM.EQ.0)ICTL=1
C TURN ON FORCE FEEDBACK
C APPLY FORCE TO TONGS
        IF(ITONGS.EQ.0)GOTO 707
        ICNTR=AB*3276.2*(D/SCL(7,1)-SCL(7,2))
     1  +(1-AB)*FLOAT(IA(8))
        CALL AOUT(11,ICNTR)
        CALL AOUT(4,-ICNTRL)
        CALL DOUT(24,"177)
C APPLY FORCE TO ARM
707     IF(IARM.EQ.0)GOTO 369
        A1=FX*R11+FY*R12+FZ*R13
        A2=FX*R21+FY*R22+FZ*R23
        A3=FX*R31+FY*R32+FZ*R33
        A(5)=A(5)+A1/SCL(1,1)
        A(7)=A(7)+A2/SCL(2,1)
        A(6)=A(6)+(A1+A3)/SCL(3,1)
        DO 766 I=5,7
        IS(I)=A(I)*3276.2+F1*(IS(I)-I3(I))
766     CONTINUE
        ICTL=0
        CALL AOUTSQ(11,17,IS)
C HAND CLOSURE TRANSFORMATIONS
C CALCULATE HAND POSITION
369     DO 124 I=2,5
        PNT(I,1)=ADAT(I,2)-1.9*D
        PNT(I,2)=ADAT(I,3)-.8*D
        PNT(I+10,1)=ADAT(I+10,2)-1.9*D
        PNT(I+10,2)=ADAT(I+10,3)-.8*D
        PNT(I+4,1)=ADAT(I+4,2)+1.9*D
        PNT(I+4,2)=ADAT(I+4,3)-.8*D
        PNT(I+14,1)=ADAT(I+14,2)+1.9*D
        PNT(I+14,2)=ADAT(I+14,3)-.8*D
124     CONTINUE
C ERASE PREVIOUS DISPLAY
        CALL LDPTRO(LOC)
C LOAD DISPLAY
C LOAD HAND INTO SUB 7
        CALL SUBDEF(7)
        DO 499 I=1,NHL
        MS=MM
        M=ACON(I,1)
        MM=ACON(I,2)
        IF(MS.EQ.M)GOTO 498
        IX=SF*ADAT(M,1)
        IY=SF*PNT(M,1)
```

```
        IZ=SF*PNT(M,2)
        CALL MOVI3(IX,IY,IZ)
498     IX=SF*ADAT(MM,1)
        IY=SF*PNT(MM,1)
        IZ=SF*PNT(MM,2)
        CALL DRWI3(IX,IY,IZ)
499     CONTINUE
        CALL SUBEND(7)
C LOAD SUBMARINE
        CALL LDTRN3(WSXX,WSXY,WSXZ,SF*WSXT,WSYX,WSYY,WSYZ,
     1  SF*WSYT)
        CALL SUBCAL(1)
C LOAD SHOULDER
        CALL LTRN(WSXX,WSXY,WSXZ,WSXT,WSYX,WSYY,WSYZ,WSYT,
     1  XX,XY,0.,0.,YX,YY,YZ,0.,ZX,ZY,ZZ,0.,SF)
        CALL SUBCAL(3)
C LOAD FOREARM
        CALL LTRN(WSXX,WSXY,WSXZ,WSXT,WSYX,WSYY,WSYZ,WSYT,
     1  XX1,XY1,XZ1,XT1,YX1,YY1,YZ1,YT1,ZX1,ZY1,ZZ1,ZT1,SF)
        CALL SUBCAL(2)
C LOAD HAND
        CALL LTRN(WSXX,WSXY,WSXZ,WSXT,WSYX,WSYY,WSYZ,WSYT,
     1  XX2,XY2,XZ2,XT2,YX2,YY2,YZ2,YT2,ZX2,ZY2,ZZ2,ZT2,SF)
        CALL SUBCAL(7)
C LOAD OBJECT
        IF(IOBJ.NE.1)GOTO 856
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  OXX,OXY,OXZ,OXT,OYX,OYY,OYZ,OYT,OZX,OZY,OZZ,OZT,SF)
        CALL SUBCAL(5)
C LOAD SHADOW
856     IF(ISHAD.NE.1)GOTO 888
C LOAD SUBMARINE SHADOW
        CALL LDTRN3(SWXX,SWXY,SWXZ,SF*SWXT,SWYX,SWYY,SWYZ,
     1  SF*SWYT)
        CALL SUBCAL(1)
C LOAD SHOULDER SHADOW
        CALL LTRN(SWXX,SWXY,SWXZ,SWXT,SWYX,SWYY,SWYZ,SWYT,
     1  XX,XY,0.,0.,YX,YY,YZ,0.,ZX,ZY,ZZ,0.,SF)
        CALL SUBCAL(3)
C LOAD FOREARM SHADOW
        CALL LTRN(SWXX,SWXY,SWXZ,SWXT,SWYX,SWYY,SWYZ,SWYT,
     1  XX1,XY1,XZ1,XT1,YX1,YY1,YZ1,YT1,ZX1,ZY1,ZZ1,ZT1,SF)
        CALL SUBCAL(2)
C LOAD HAND SHADOW
        CALL LTRN(SWXX,SWXY,SWXZ,SWXT,SWYX,SWYY,SWYZ,SWYT,
     1  XX2,XY2,XZ2,XT2,YX2,YY2,YZ2,YT2,ZX2,ZY2,ZZ2,ZT2,SF)
        CALL SUBCAL(7)
C LOAD OBJECT SHADOW
        IF(IOBJ.NE.1)GOTO 888
```

```
        CALL LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1  0XX,0XY,0XZ,0XT,0.,0.,0.,-48.75,0ZX,0ZY,0ZZ,0ZT,SF)
        CALL SUBCAL(5)
888     CALL LDTRN3(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT)
C DISPLAY CENTER OF HAND
        IF(ITCHPT.NE.1)GOTO 130
        IX=SF*XC
        IY=SF*YC
        IZ=SF*ZC
        CALL MOVI3(IX,IY,IZ)
        CALL DRWI3(IX,IY,IZ)
C DISPLAY SHADOW
        IF(ISHAD.NE.1)GOTO 130
        IX=SF*XC
        IY=-1950
        IZ=SF*ZC
        CALL MOVI3(IX,IY,IZ)
        CALL DRWI3(IX,IY,IZ)
C LOAD PERMANANT DISPLAY
130     CALL SUBCAL(6)
C DISPLAY CROSS-SECTION OF SURFACE UNDER HAND
        IF(ISURF.NE.1)GOTO 843
        CALL SETINT(15)
        IY=SF*YSURF(2000./SF,ZT,P1,P2)
        CALL MOVI3(IFIX(X),IY,IFIX(SF*ZT))
        DO 808 I=-2000,2000,80
        X=I
        IY=SF*YSURF(X/SF,ZT,P1,P2)
        IF(DEP.GT.0)IY=FLOAT(IY)-SF*DEP*
     1  EXP(-.004*(1.-BC)*(X/SF-XT)**2.)
        CALL DRWI3(I,IY,IFIX(SF*ZT))
808     CONTINUE
C DISPLAY PROXIMITY INDICATOR
843     IF(IPROX.NE.1)GOTO 699
        CALL LDTRN3(SF,0.,0.,0.,0.,SF,0.,0.)
        CALL MOVI3(0,1000,0)
        IF(PROX.GT.24)PROX=24.+(PROX-24.)/10.
        CALL RDRWI3(0,IFIX(40.*PROX),0)
        CALL MOVI3(-100,1000,0)
        CALL DRWI3(100,1000,0)
C DISPLAY SPHERE
699     IF(ISPH.NE.1)GOTO 340
        IX=SF/S*(WXX*XO+WXZ*ZO)
        IY=SF/S*(WYX*XO+WYY*YO+WYZ*ZO)
        IZ=SF/S*(WZX*XO+WZY*YO+WZZ*ZO)
        CALL LDTRN3(S,0.,0.,WXT,0.,S,0.,WYT)
        CALL MOVI3(IX,IY,IZ)
        CALL SUBCAL(4)
C DISPLAY SPHERE SHADOW
```

```
        IF(ISHAD.NE.1)GOTO 340
        IX=SF*XO
        IY=SF*ZO
        IZ=-1950
        CALL LDTRN3(WXX,WXZ,WXY,WXT,WYX,WYZ,WYY,WYT)
        CALL MOVI3(IX,IY,IZ)
        CALL SUBCAL(4)
C SEND PICTURE TO DISPLAY BUFFER
340     CALL MGSEND
C READ KEYBOARD
        IF(IFF.NE.1)CALL QIO("10400,5,3,,IOSB,IPARAM,IDS)
        IFF=1
        CALL READEF(3,IUU)
        IF(IUU.NE.2)GOTO 112
C CLEAR ENTRY
        WRITE(5,999)"033,"114,"033,"101
999     FORMAT('+',4A)
        IFF=0
C DECODE INPUT
        IF(ICMD.EQ."070)DWA2=DWA2+.01
        IF(ICMD.EQ."062)DWA2=DWA2-.01
        IF(ICMD.EQ."064)DWA1=DWA1-.01
        IF(ICMD.EQ."066)DWA1=DWA1+.01
        IF(ICMD.EQ."075)DTY=DTY+40.
        IF(ICMD.EQ."134)DTX=DTX+40.
        IF(ICMD.EQ."047)DTY=DTY-40.
        IF(ICMD.EQ."133)DTX=DTX-40.
        IF(ICMD.NE."065)GOTO 711
        DWA1=0
        DWA2=0
        DS=0
        DTX=0
        DTY=0
711     IF(ICMD.EQ."071)DS=DS+.01
        IF(ICMD.EQ."063)DS=DS-.01
        IF(ICMD.EQ."132)STOP
        IF(ICMD.EQ."123)ISHAD=-ISHAD
        IF(ICMD.EQ."127)IWALL=-IWALL
        IF(ICMD.EQ."124)ITCHPT=-ITCHPT
        IF(ICMD.EQ."120)IPROX=-IPROX
        IF(ICMD.EQ."122)GOTO 100
        IF(ICMD.NE."104)GOTO 600
        IX=SF*XT
        IY=SF*YT
        IZ=SF*ZT
        CALL MODIFY(LINES)
        CALL DRWI3(IX,IY,IZ)
        LINES=LINES+2
        IF(ISHAD.EQ.1)GOTO 600
```

```
        CALL MODIFY(LSHADS)
        CALL DRWI3(IX,-1950,IZ)
        LSHADS=LSHADS+2
600     IF(ICMD.NE."115)GOTO 601
        IX=SF*XT
        IY=SF*YT
        IZ=SF*ZT
        CALL MODIFY(LINES)
        CALL MOVI3(IX,IY,IZ)
        LINES=LINES+2
        IF(ISHAD.EQ.1)GOTO 601
        CALL MODIFY(LSHADS)
        CALL MOVI3(IX,-1950,IZ)
        LSHADS=LSHADS+2
601     IF(ICMD.NE."105)GOTO 602
        LINES=LINES-2
        CALL MODIFY(LINES)
        CALL RMOVI3(0,0,0)
        IF(ISHAD.NE.1)GOTO 602
        LSHADS=LSHADS-2
        CALL MODIFY(LSHADS)
        CALL RMOVI3(0,0,0)
602     IF(ICMD.NE."067)GOTO 603
        WA1=1.57
        WA2=0.
603     IF(ICMD.NE."061)GOTO 604
        WA1=0
        WA2=0
604     IF(ICMD.NE."060)GOTO 112
        TX=0
        TY=0
        GOTO 112
        END
        FUNCTION YSURF(X,Z,P1,P2)
        YSURF=(SIN(.1256*Z)+COS(.02472*Z))*P1*X/(P2+X**2)-35.
        RETURN
        END
        SUBROUTINE DRAW(IPNTS,SCL)
        DO 200 I=1,IPNTS
        MS=MM
C READ CONECTIVITY DATA
        READ(1,*,END=300)M,MM
C IF STARTING POINT WAS LAST LOCATION SKIP MOVE
        IF(M.EQ.MS)GOTO 100
C READ COORDINATES AND DRAW LINE
        READ(2'M)X,Y,Z
        IX=SCL*X
        IY=SCL*Y
        IZ=SCL*Z
```

```
      CALL MOVI3(IX,IY,IZ)
100   READ(2'MM)X,Y,Z
      IX=SCL*X
      IY=SCL*Y
      IZ=SCL*Z
      CALL DRWI3(IX,IY,IZ)
200   CONTINUE
300   RETURN
      END
      SUBROUTINE LTRN(WXX,WXY,WXZ,WXT,WYX,WYY,WYZ,WYT,
     1 XX,XY,XZ,XT,YX,YY,YZ,YT,ZX,ZY,ZZ,ZT,SF)
      CALL LDTRN3(WXX*XX+WXY*YX+WXZ*ZX,WXX*XY+WXY*YY+WXZ*ZY,
     1 WXX*XZ+WXY*YZ+WXZ*ZZ,SF*(WXX*XT+WXY*YT+WXZ*ZT+WXT),
     1 WYX*XX+WYY*YX+WYZ*ZX,WYX*XY+WYY*YY+WYZ*ZY,WYX*XZ
     1 +WYY*YZ+WYZ*ZZ,SF*(WYX*XT+WYY*YT+WYZ*ZT+WYT))
      RETURN
      END
```

APPENDIX VII

MANIPULATOR CONTROL SOFTWARE

The control interface is able to independently designate each degree of freedom of the manipulator to be computer control or master-slave. This is done by sending a 16 bit control word to the interface through digital output port 24. The bits correspond to the degrees-of-freedom as follows:

```
0/TAR/LZY/XOT/ARL/ZYX
|--SLAVE--+--MASTER--|
```

The bits are set to one for computer control. The value of each degree-of-freedom is read through analog input and controlled through analog output via the channels shown in table 7.

TABLE 7 CONTROL INTERFACE I/O CHANNELS

DEGREE OF FREEDOM		ANALOG INPUT CHANNEL	ANALOG OUTOUT CHANNEL
	T	16	4
S	A	17	5
L	R	18	6
A	L	19	7
V	Z	20	8
E	Y	21	9
	X	22	10
M	T	23	11
A	A	24	12
S	R	25	13
T	L	26	14
E	Z	27	15
R	Y	28	16
	X	29	17

The manipulator angles used in this simulation are

linear combinations of the above degrees-of-freedom. The relationships between the two are given by:

A1=Z
A2=X
A3=Y-Z
A4=A
A5=(R-L)/2
A6=(R+L)/2
D=T

The following program was used to control the manipulator:

CNTRL.FTN

```
C THIS PROGRAM ALLOWS THE MASTER MANIPULATOR TO MOVE
C FREELY UNDER COMPUTER CONTROL WHEN ICTL=1, NO
C ACTION IS TAKEN WHEN ICTL=0. THE CYCLE TIMES (IT1,IT2),
C  DEGREE OF TACH FORWARD (F1) AND CONTROL WORD (ICTRL)
C  ARE INPUT THROUGH DMABUF.
      COMMON /DMABUF/IDUM(3894),II(7),I1(7),I2(7),I3(7),
     1  IS(7),F1,IT1,IT2,ICTL
      ICTL=0
C CHECK TO DETERMINE IF CONTROL DESIRED
100   IF(ICTL.EQ.0)GOTO 800
C SAVE LAST POSITION FOR TACH FORWARD
      DO 200 I=1,6
      I2(I)=I1(I)
200   CONTINUE
C READ MANIPULATOR POSITION
      CALL AINSQ(24,29,I1)
C ADD IN TACH FORWARD
      DO 300 I=1,6
      II(I)=FLOAT(I1(I))+F1*FLOAT(I1(I)-I2(I))
300   CONTINUE
C SEND NEW MANIPULATOR POSITION
      CALL AOUTSQ(12,17,II)
C WAIT FOR NEXT CYCLE
700   CALL WAIT(IT1,1,IDS)
      GOTO 100
800   CALL WAIT(IT2,1,IDS)
      GOTO 100
      END
```

APPENDIX VIII

PROGRAM DATA BASES

SCALE.DAT

This file contains scaling factors for scaling A/D output to manipulator angles.

SCALE FACTOR	OFFSET
.082	.5
.082	-.0
.196	-8.1575
-.314	0.
-.314	-2.
-.32	5.
.33	-.9865

ARMDAT.DAT

This file contains manipulator point data. The point number is given for reference only, it is not part of the file.

POINT #	X	Y	Z
1	.5	1.3	-1.3
2	.5	3.3	-2.5
3	.5	2.4	-5.5
4	.5	1.9	-5.5
5	.5	1.9	-3.5
6	.5	-1.9	-3.5
7	.5	-1.9	-5.5
8	.5	-2.4	-5.5
9	.5	-3.3	-2.5
10	.5	-1.3	-1.3
11	-.5	1.3	-1.3
12	-.5	3.3	-2.5
13	-.5	2.4	-5.5
14	-.5	1.9	-5.5
15	-.5	1.9	-3.5
16	-.5	-1.9	-3.5
17	-.5	-1.9	-5.5
18	-.5	-2.4	-5.5
19	-.5	-3.3	-2.5
20	-.5	-1.3	-1.3
21	0.	0.	0.
22	0	0	-40.3
23	1.	1	-1.3

POINT #	X	Y	X
24	1.	1	-39
25	1.	-1	-39
26	1.	-1	-1.3
27	-1.	1	-1.3
28	-1.	1	-39
29	-1.	-1	-39
30	-1.	-1	-1.3
31	0.	0.	0.
32	0	0	-18.4
33	1.	1	0
34	1.	1	-17.1
35	1.	-1	-17.1
36	1.	-1	0
37	-1.	1	0
38	-1.	1	-17.1
39	-1.	-1	-17.1
40	-1.	-1	0

ARMCON.DAT

This file contains manipulator connectivity data. Lines are draw between starting point (SP) and endpoint (EP) defined in ARMDAT.DAT

SP	EP	SP	EP	SP	EP
1	2	2	3	3	4
4	5	5	6	6	7
7	8	8	9	9	10
10	1	11	12	12	13
13	14	14	15	15	16
16	17	17	18	18	19
19	20	20	11	1	11
2	12	3	13	4	14
5	15	6	16	7	17
8	18	9	19	10	20
21	1	21	11	21	10
21	20	22	24	22	28
22	25	22	29	23	24
24	25	25	26	26	23
27	28	28	29	29	30
30	27	23	27	24	28
25	29	26	30	31	23
31	27	31	26	31	30
32	34	32	38	32	35
32	39	33	34	34	35
35	36	36	33	37	38

SP	EP	SP	EP	SP	EP
38	39	39	40	40	37
33	37	34	38	35	39
36	40				

SUBDAT.DAT

This file contains submarine point data.

POINT #	X	Y	Z
1	0	12.5	-10
2	0	15	0
3	5	12.5	0
4	7.5	7.5	0
5	5	2.5	0
6	0	0	0
7	-5	2.5	0
8	-7.5	7.5	0
9	-5	12.5	0
10	0	15	25
11	5	12.5	25
12	7.5	7.5	25
13	5	2.5	25
14	0	0	25
15	-5	2.5	25
16	-7.5	7.5	25
17	-5	12.5	25
18	0	7.5	40
19	0	12.5	42
20	0	12.5	45
21	0	2.5	45
22	0	2.5	42
23	5	7.5	42
24	5	7.5	45
25	-5	7.5	45
26	-5	7.5	42
27	5	12.5	5
28	5	12.5	10
29	0	15	15
30	-5	12.5	10
31	-5	12.5	5
32	0	20	2
33	3.5	20	5
34	3.5	20	10
35	0	20	13
36	-3.5	20	10
37	-3.5	20	5
38	0	7.5	44

SUBCON.DAT

This file contains submarine connectivity data.

SP	EP	SP	EP	SP	EP
1	2	1	3	1	4
1	5	1	6	1	7
1	8	1	9	29	10
3	11	4	12	5	13
6	14	7	15	8	16
9	17	10	18	11	18
12	18	13	18	14	18
15	18	16	18	17	18
2	3	3	4	4	5
5	6	6	7	7	8
8	9	9	2	10	11
11	12	12	13	13	14
14	15	15	16	16	17
17	10	18	19	19	20
20	38	38	21	21	22
22	18	18	23	23	24
24	38	38	25	25	26
26	18	18	38	2	27
27	28	28	29	29	30
30	31	31	2	32	33
33	34	34	35	35	36
36	37	37	32	2	32
27	33	28	34	29	35
30	36	31	37		

OBJDAT.DAT

This file contains object point data for a rectangular peg.

POINT #	X	Y	Z
1	-1	.5	-1
2	1	.5	-1
3	1	-.5	-1
4	-1	-.5	-1
5	-1	.5	1
6	1	.5	1
7	1	-.5	1
8	-1	-.5	1
9	0	0	1
10	0	0	3

OBJCON.DAT

This file contains connectivity data for the rectangular peg.

SP	EP	SP	EP	SP	EP
1	2	2	3	3	4
4	1	5	6	6	7
7	8	8	5	1	5
2	6	3	7	4	8
9	10				

OBJTCH.DAT

This file contains touching conditions for the rectangular peg. These conditions are rectanqular with center (Xc,Yc, Zc) and dimensions (A,B,C).

Xc	Yc	Zc	A	B	C
0	0	0	1	.5	1
0	0	2	.1	.1	1